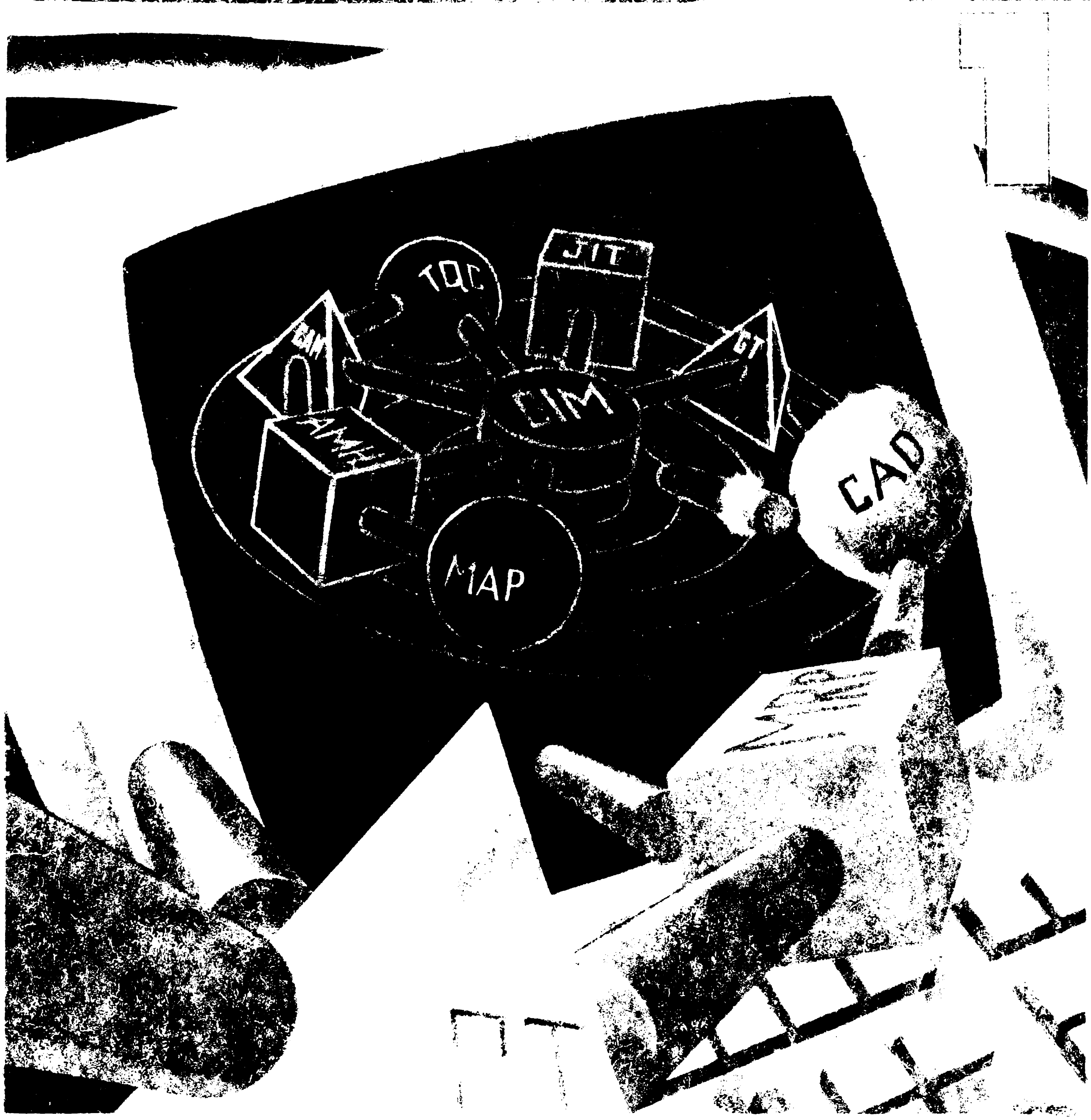
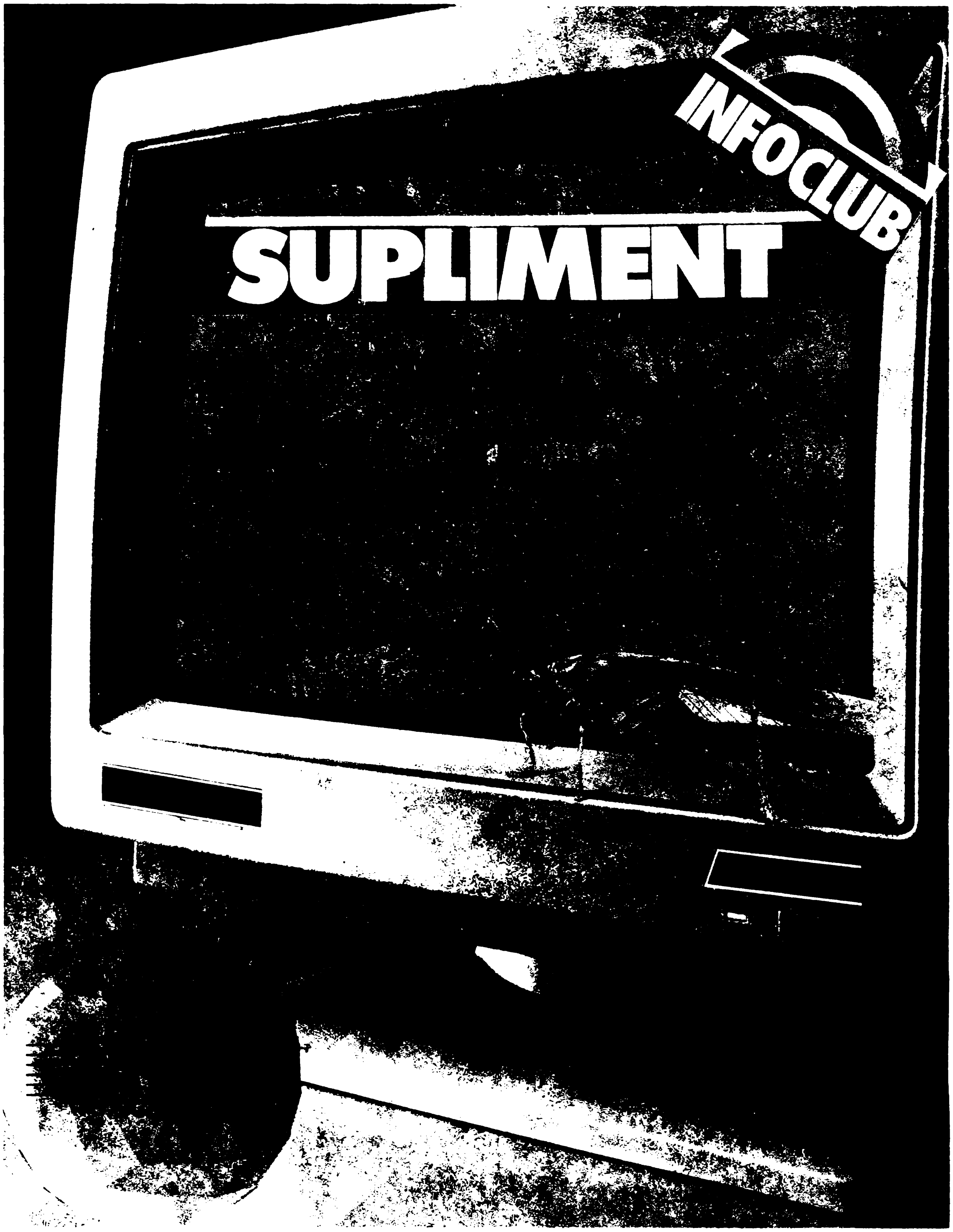
SUPLIMENT
DE INFORMATICA SI CALCULATOARE





INTERNATIONAL DATA GROUP

The World's Leader in Information Services on Information Technology

În sfîrșit...

...Sîntem în măsură să vă oferim mult așteptatul supliment de informatică și calculatoare intitulat — INFOCLUB!

De ce „INFOCLUB"?
Foarte simplu! Deoarece acest supliment nu apare pe un teren arid, el continuă, în mod firesc, rubrica cu același nume, din revista „Știință și tehnică" care a promovat în mod constant informatica, sub diverse rubrici, oferind cititorilor și pasionaților informaticii o gamă cît mai variată de date și informații dintr-un domeniu de foarte mare interes.

Datorită nenumăratelor scrisori primite de la dumneavoastră, dragi cititori, rubricile de informatică au căutat — în limitele spațiului de care a dispus și a unor condiții „obiective" asupra cărora nu mai revenim — să acopere din cît mai multe unghiuri problemele calculatoarelor. Așa au apărut rubricile: **„Un limbaj de programare în serial", „Școala la ora informaticii", „Infoclub", „Programe comentate", „Consultații BASIC"**, etc., care s-au bucurat de un deosebit succes.

Acesta este și motivul pentru care vom publica în acest prim număr al suplimentului nostru, ceea ce noi am intitulat **„Bibliografia ST de informatică"** pe ultimii 4 ani (fără ca aceasta să fie răspunsul complet la întrebarea: „ce am făcut în ultimii 5 ani?") pentru a veni în întîmpinarea tuturor iubitorilor informaticii din țară, contribuind, cu mijloace proprii, la instruirea lor.

Ce își propune „INFOCLUB"? sau **Cui se adresează INFOCLUB?**
Situația obiectivă a industriei (hard și soft) de profil din țara noastră, cît și cerințele reale ale începătorilor dar și ale celor care doresc să se perfecționeze, au condus la structurarea acestui supliment în trei părți distincte: o bună parte (aproape jumătate) este destinată compatibilelor Spectrum dată fiind răspîndirea acestui tip de „home computer" în țara noastră; un spațiu însemnat este acordat problemelor legate de calculatoarele personale profesionale și, în sfîrșit, am rezervat un număr de pagini și unor aplicații pentru microcalculatoarele compatibile CP/M.

Desigur că nu lipsesc noutățile de ultimă oră din domeniu, tendințe actuale de piață, informații la zi despre cluburi și societăți de informatică care au fost înființate în țara noastră, toate acestea întregind imaginea unui domeniu foarte dinamic. Scopul principal a întregului supliment este, după cum veți vedea, prezentarea aplicațiilor concrete și nu a teoriei, astfel încît, după parcurgerea acestuia, cititorul să posede un bagaj sporit de cunoștințe în capitolul care îl interesează. Sperăm că am reușit să răspundem preferințelor dv., în orice caz, vă așteptăm cu sugestiile — așa cum ați făcut-o în toți acești ai pentru rubrica de informatică din revistă — astfel încît conținutul suplimentului să vină în întîmpinarea preferințelor și dorințelor tuturor iubitorlor informaticii.

Dorim să încheiem, dragi prieteni, cu mulțumirile noastre pentru atenția cu care ați susținut permanent rubricile de informatică din revista „Știință și tehnică", cu credința că, în viitor, vom colabora la fel de strîns pentru a recupera în cel mai scurt timp inerțiile obiective ce ne-au grevat activitatea în anii ce au trecut!

Mihaela Gorodcov

CUPRINS
CONTENTS



Valeriu IORGA

Născut în Lugoj, în anul 1943. Absolvent al Facultăţii de Energetică, secţia Automatică.
Hobby: muzica, literatura SF, cuvinte încrucişate.

Repartizat, după absolvirea facultăţii, în anul 1966, ca asistent preparator la catedra de calculatoare a facultăţii de Automatică din cadrul Institutului Politehnic Bucureşti. Ocupă prin concurs posturile de asistent titular (1971), şef de lucrări (1975) şi conferenţiar (1990).

În anul 1982 a obţinut titlul ştinţific de doctor inginer, cu o lucrare din domeniul conducerii numerice directe a proceselor tehnologice. În toată această perioadă desfăşoară activitate didactică de predare de cursuri în domeniul calculatoarelor cum ar fi: „Programarea calculatoarelor“, „Calcul numeric“ şi „Tehnici de programare“.

În activitatea de cercetare are preocupări legate de sisteme de operare şi aplicaţii în timp-real, teoria algoritmilor.

A desfăşurat o bogată activitate publicistică, avînd tipărite 11 cărţi şi manuale precum şi numeroase articole în domeniul programării calculatoarelor, inclusiv în revista „Ştiinţă şi tehnică“. Pentru viitor aşteptăm cu încredere noi apariţii editoriale în domeniul calculatoarelor şi a aplicaţiilor acestora.



Doina ISTRĂTESCU

Născută în Bucureşti, în anul 1954. Absolventă a Facultăţii de Automatică, secţia calculatoare.
Hobby: baletul şi muzica clasică.

Nu după mulţi ani de la absolvirea facultăţii, alături de foarte mulţi tineri dar şi specialişti cu experienţă, s-a integrat în rîndul cercetătorilor din cadrul Institutului de Tehnică de Calcul din Bucureşti. În cei 11 ani de cînd îşi desfăşoară activitatea în colectivul acestui institut şi-a orientat activitatea de cercetare ştiinţifică, în principal, în domenii de specialitate cum ar fi: informatica medicală, baze de date, grafică şi proiectare asistată de calculator ş.a.

Nefiind la prima „încercare“ publicistică sperăm într-o colaborare număr la număr la acestă nouă apariţie editorială intitulată atît de sugestiv „Infoclub“.



Virgil IONESCU

Născut în Bucureşti, în anul 1951. Absolvent al Facultăţii deFizică.
Hobby: pescuitul, fotografia, electronica, filatelia şi, bineînţeles, calculatorul.

Profesor de fizică la Şcoala nr.25 din Bucureşti şi-a început activitatea legată de această mare pasiune — informatica — printr-o colaborare la primul număr al „Buletinului de informatică“ editat de Inspectoratul Şcolar al municipiului Bucureşti, la revista „Învăţămîntul liceal şi tehnic profesional“, precum şi la sesiunile de comunicări ştiinţifice ale „Societăţii de ştiinţe matematice“ — secţiunea de informatică. Ulterior, a trecut de la simpla colaborare la redactarea (împreună şi cu alţi profesori) a buletinului de informatică al inspectoratului şcolar din capitală.

Pentru viitor îşi propune editarea unei reviste de informatică orientată spre problematica şcolară.



Călin OBRETIN

Născut în Bucureşti, în anul 1958. Absolvent al Facultăţii de Geodezie.
Hobby: Karate — antrenor centura neagră, literatura şi filmele SF.

În tot acest timp, are o pasiune constantă, calculatorul, căruia îi dedică cea mai mare parte a timpului. Cu mult entuziasm, alături de un grup de „fani“, în anul 1987, înfiinţează în cadrul Casei de Cultură a studenţilor, Clubul de calculatoare din Bucureşti, deschis tuturor posesorilor de „personale“ din Bucureşti şi din ţară. În ianuarie 1990, Clubul de calculatoare devine Clubul Român de Calculatoare (după cum reiese şi din materialul pe care îl prezentăm în acest supliment) condus şi animat de acelaşi Călin Obretin. În intenţiile viitoare ale clubului: contactarea cluburilor similare din străinătate (au fost deja realizate contacte foarte promiţătoare), editarea unei reviste a clubului (noi, în mod sincer, sportiv, le dorim succes!), punerea la punct a unei reţele şi multe altele.

Curriculum vitae



Liviu DUMITRAŞCU

Născut în România cu un număr de ani în urmă!! Absolvent al Facultăţii de Mecanică Fină.
Hobby: muzica, atît ca ascultător, cît şi ca interpret.

După absolvirea facultăţii, timp de 10 ani activează în cadrul CEPECA unde predă cursuri privind limbaje de programare a calculatoarelor.

După absolvirea în Franţa a unor cursuri de specializare în informatică şi tehnică de calcul îl găsim ca şef de lucrări în cadrul Institutului de Petrol şi Gaze din Ploieşti unde îşi desfăşoară activitatea şi în prezent. Între timp, în anul 1986 îşi susţine şi teza de doctorat în domeniul automatizării instalaiilor petroliere.

Activitatea publicistică, în calitate de autor sau coautor s-a materializat în cadrul Editurii Academiei şi Editurii Tehnice prin numeroase articole şi prin şase volume care s-au bucurat de o mare popularitate, unul dintre acestea fiind editat şi în limba engleză. Iată cîteva titluri: „Învăţăm microelectronica interactivă", „Tehnici de construire a programelor cu structuri alternative", „Elemente de inteligenţă artificială pentru conducerea operativă a producţiei".

Are numeroase proiecte de viitor pe care însă nu le vom dezvălui deocamdată. Poate într-un număr viitor al suplimentului nostru de informatică şi calculatoare!



Stelian NICULESCU

Născut în comuna Stoeneşti, jud. Olt, în anul 1937. Absolvent al Facultăţii de Matematică.
Hobby: umorul şi... şahul.

Doctor matematician, binecunoscut de peste 25 de ani informaticienilor din ţară, autor a numeroase lucrări, circa 14 volume apărute în Editura Didactică şi Pedagogică, Editura Ştiinţifică şi Enciclopedică, precum şi în Editura Tehnică, duce de peste două decenii o susţinută activitate didactică şi de cercetare atît în învăţămîntul preuniversitar cît şi în cel universitar.

Activitatea de cercetare ştiinţifică şi publicistică s-a materializat în peste 20 de lucrări de cercetare publicate în ţară şi în străinătate referitoare la limbaje de programare, instruire asistată de calculator, cercetări operaţionale ş.a. Sperăm într-o activitate publicistică viitoare pe măsura numelui şi a experienţei sale didactice atît în paginile revistei „Ştiinţă şi tehnică" dar mai ales în paginile acestui supliment de informatică şi calculatoare care se a.reşte pe viitor într-o apariţie trimestrială la îndemîna tuturor iubitorilor acestui domeniu mirific al ştiinţei.



Eugen GEORGESCU

Născut în Bucureşti, în anul 1957. Absolvent al Facultăţii deElectronică şi Telecomunicaţii.
Hobby: fotografia, muzica progresivă şi... speologia.

Imediat după terminarea facultăţii a lucrat un număr de ani în domeniul exploatării emiţătoarelor de Radio-TV, apoi în proiectarea acestora. Pasiunea nedeclarată însă pentru informatică şi calculatoare a fost mai puternică, fapt pentru care, în ultimii cinci ani, îl găsim în rîndul cercetătorilor Institutului de Tehnică de Calcul din Bucureşti.

Specializarea în domenii atît de importante ale informaticii cum ar fi bazele de date, grafică şi proiectare asistată de calculator, reţele de calculatoare şi sisteme de operare, dublată de un real talent publicistic îi deschid un cîmp larg de acţiune pe această linie. Ca urmare, în numerele viitoare ale suplimentului de informatică şi calculatoare al revistei „Ştiinţă şi tehnică", sperăm ca împreună cu colega sa de institut şi nu numai... să vadă lumina tiparului o serie de alte materiale publicistice legate de limbaje neprocedurale în baze de date, algoritmi de grafică precum şi un minicurs de „C".

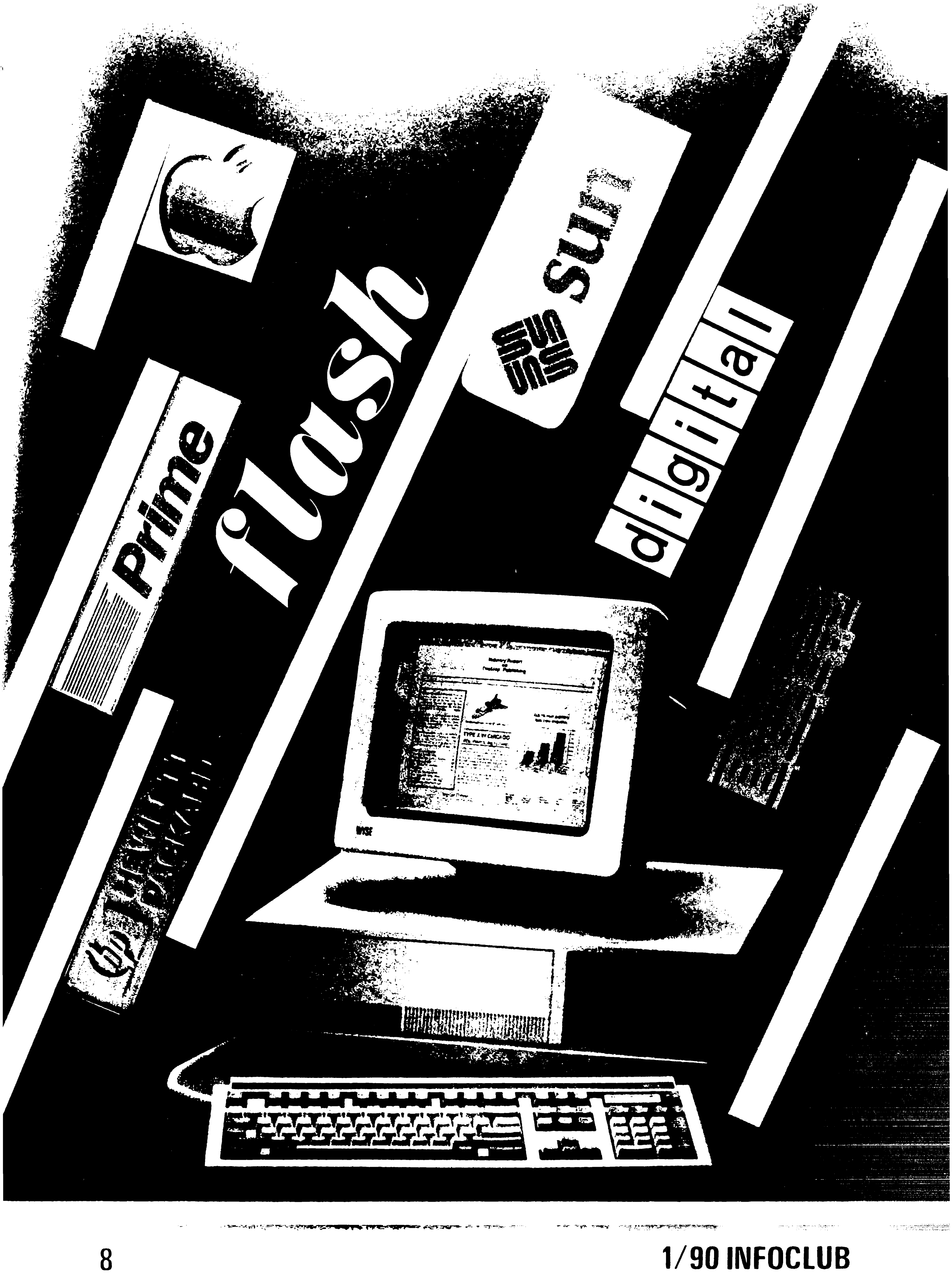


Ion DIAMANDI

Născut în Bucureşti, în anul 1951. Absolvent al Facultăţii deCibernetică.
Hobby: tenis şi bridge.

După absolvirea facultăţii, în anul 1975, a activat în cadrul Institutului de Tehnică de Calcul din Bucureşti, conducînd un colectiv cu preocupări în domeniul calculatoarelor personale şi o temă de cercetare privind utilizarea calculatoarelor în rocesul de educaţie.

Are o intensă activitate publicistică în domeniu, fiind prezent prin o multitudine de articole în paginile revistelor de popularizare a ştiinţei pentru copii şi tineret, deci şi a revistei „Ştiinţă şi tehnică", fiind în acelaşi timp o prezenţă activă în numeroasele cercuri de calculatoare pentru copii. Este autor sau coautor al unor lucrări din care amintim cîteva titluri mai semnificative: „Diaiog cu viitorul", „Partenerul meu de joc — calculatorul", „Elemente de informatică pentru cercurile tehnico-aplicative", „Buletin metodic de informatică" ş a.


Prime
flash
sun
digital

Ion
DIAMANDI

Stații de lucru inginerești

Stațiile de lucru inginerești reprezintă un domeniu relativ nou, apărut după cel al calculatoarelor personale din care s-a desprins. În categoria stațiilor de lucru inginerești sînt cuprinse stațiile grafice, echipamentele pentru: proiectarea asistată de calculator (CAD), fabricația asistată de calculator (CAM), ingineria asistată de calculator (CAE) etc. O dată cu apariția și producerea calculatoarelor personale de mare performanță, stațiile de lucru inginerești cunosc și ele o mare dezvoltare.

De remarcat faptul că în anul 1986 vînzările de stații de lucru inginerești au depășit pe cele ale calculatoarelor personale, piața SUA în acest an înregistrînd o creștere de 40%. De altfel, precedentul fusese creat: în anul 1985 fuseseră deja vîndute în SUA 21 000 de stații de lucru inginerești care reprezentau o valoare de 735 milioane dolari. De fapt drumul stațiilor de lucru inginerești fusese schimbat de firma Apollo în anul 1981 care, ulterior, a și dominat această piață cu un procent de 30—40% din vînzări. În anul 1984, însă, și-a făcut o apariție meteorică firma SUN care, în perioada 1985-1986, a preluat conducerea acestei piețe.

Deoarece tendințele arată că bătălia viitoarelor calculatoare se va da între calculatoarele de birou și stațiile de lucru inginerești se cuvine să se facă o analiză mai amănunțită a acestora, prin aflarea răspunsurilor la o serie de întrebări:

- **ce sînt sau ce se înțelege, de fapt, prin stații de lucru inginerești?**
- **care sînt producătorii de stații de lucru inginerești?**
- **care este situația pieței?**
- **care sînt caracteristicile care fac ca stațiile de lucru să se constituie într-o formă unică de prelucrare a informațiilor?**
- **care sînt aplicațiile ce se pretează abordărilor cu stații de lucru inginerești?**

Iată o posibilă definiție a unei stații de lucru inginerești bazată în primul rînd pe tehnologie și componență: unitate centrală pe 32 de biți (sau mai mult); memorie internă de 2-4 Mo și, de obicei, suport de memorie virtuală; display integral cu o rezoluție de minim 640 x 480 pixeli; interfețe pentru rețea (de obicei Ethernet); sistem de operare multitasking (nefiind neapărat necesar un sistem de operare multiutilizator, deși unele stații prezintă acest tip de sistem).

Există firește și alte definiții care nu exclud elementele din lista expusă, permițînd stației să prelucreze unul sau mai multe taskuri tipice. O dimensiune mare a registrelor memoriei interne precum și un spațiu de adresare suplimentar (eventual virtual) sînt necesare pentru rularea unor aplicații tipice inginerești. Rezoluția înaltă depinde direct de magistrala microprocesorului și ea permite o interfață cu utilizatorul prin ferestre, ceea ce are ca urmare creșterea productivității muncii la utilizarea stației. Ușurința integrării într-o rețea este prima armă împotriva formării unor „insule de prelucrare informațională" și, în sfîrșit, sistemul de operare multitasking leagă toate aceste elemente împreună.

Exemple tipice de mașini care se potrivesc definiției date sînt stațiile care aparțin liniei VAX a firmei Digital Equipment (DEC) precum și produsele cele mai utilizate ale producătorilor tradiționali ca SUN și Apollo. De remarcat că există mai multe firme (InterAct, InterPro) unite sub numele de Intergraph care produc stații pe bază de VAX sau MicroVax și care dețineau împreună în anul 1988 circa 29% din piața stațiilor de lucru.

DEC a intrat pe piața stațiilor de lucru grafice în anul 1984 cu prima stație VAX pentru satisfacerea necesităților impuse de aplicațiile CAD/CAM/CAE. Deși cu performanțe ceva mai scăzute și costuri mai ridicate, linia de stații VAX a cîștigat teren în mod continuu, devenind a treia putere după SUN și Apollo. În anii 1985-1986 modelele VAX station II și VAX GPX reprezentau unele din cele mai căutate produse pe piața mondială oferind, în afara facilităților grafice specifice domeniului și celelalte programe aplicative funcționînd sub sistemul de operare VMS (după unele estimări, DEC a livrat în 1985 echipamente CAD/CAM pentru circa 1/3 din piața mondială). De fapt, în acea perioadă, avîndu-se în vedere faptul că producătorii declarau mărirea capacității și performanțelor grafice, se prognoza pentru stațiile de lu-

cru cu grafică în 3D o creştere de 60% pentru 1988. Dar, Charles D. Wise, software manajer pentru VAX/VAM, a prevăzut un conflict între staţiile VAX şi calculatoarele VAX multiutilizator: „DEC are un interes vast la mini şi supermini şi, din această cauză, nu are de gînd să facă o afacere din staţiile de lucru care ar reduce vînzările pentru VAX 8 000". Urmarea a fost o scădere simţitoare a procentului pe piaţa staţiilor de lucru deţinută de DEC. Deci, la sfîrşitul anului 1987 piaţa staţiilor de lucru inginereşti se prezenta astfel: Sun Microsystems —33%; Apollo — 18%; DEC — 3%; HP— 2%.

Iată şi prognoza Dataguest la acea vreme în ceea ce priveşte performanţele staţiilor (viteză de lucru): 1987. 7-10 MIPS; 1988: 16-20 MIPS; 1989: 30-40 MIPS. Un exemplu de staţie DEC la nivelul anilor 1987-1988: staţia VAX 3 200 cu performanţe de 10 MIPS.

Pe piaţa staţiilor de lucru au apărut şi producători mai puţin tradiţionali. Astfel, Hewlett Packard care a venit din domeniul minicalculatoarelor, precum şi Tektronix, din domeniul terminalelor, sînt două exemple. Hewlett Packard de mai mulţi ani are un rol, deloc neglijabil pe piaţa staţiilor de lucru. Deşi a vîndut un număr mare de unităţi, unele din acestea nu intră în definiţia dată anterior staţiilor. Astfel, deşi a vîndut bine seria 9 000 (care se potriveşte standardului, performanţelor şi preţului), Hewlett Packard rămîne mai mult un furnizor de nişă pe piaţa staţiilor de lucru, majoritatea acestora intrînd în categoria sistemelor la cheie. Tektronix a început să fabrice a doua sa serie de staţii de lucru. Prima încercare (seriile 61XX şi 62XXs) a fost o „semireuşită" prezentînd cîteva probleme atît de producţie — cu circuitul integrat National 32 000 — cît şi cu software-ul. Concepţia firmei este că piaţa tradiţională de terminale grafice se va transforma rapid în piaţă de staţii de lucru. Date fiind mărimea şi forţa sa financiară Tektronix rămîne o firmă a cărei evoluţie pe piaţa staţiilor de lucru trebuie urmărită.

Un alt producător important este, desigur, IBM. Însă singurul său produs care se potriveşte definiţiei date este staţia 6150 PC/RT (microcalculator de tip RISC cu set redus de instrucţiuni), cu toate că şi PS/2 Model 80 cu sistemul de operare OS/2 şi Model 55 cu microprocesor 80 386 SX pot fi considerate adevărate staţii de lucru. Deşi a făcut îmbunătăţiri staţiei 6150 PC/RT, aceasta este departe de a reprezenta un succes răsunător. Staţiile PC/RT sînt declarate de firmă echipamente profesionale pe 32 de biţi, cu sistem IBM/AIX, preţul lor fiind destul de mare mai ales prin faptul că sînt realizate cu microprocesoare IBM originale. Iată şi membrii familiei 6150 PC/RT împreună cu preţurile corespunzătoare anilor 1988-1989: IBM PC RT Model 1 (desktop) — 13 000 dolari; IBM PC RT Model 2 (floor-standing) — 40 000 dolari; IBM PC RT 130 (desk top) — 23 000 dolari; IBM PC RT 135 (floor standing) — 30 000 dolari; IBM PC B 35 (+ IBM 5080 Graphics) — 32 000 dolari.

Cu toată această gamă variată PS/2 Model 80 şi mult discutatul Model 90 se încadrează mai bine în definiţia dată staţiilor de lucru. Dată fiind baza instalată de PC-uri, aceste maşini pot muta IBM-ul într-o categorie foarte serioasă de producători de staţii. De asemenea, calculatorul personal MAC II cu sistemul de operare A/UX poate aduce şi firma Apple în lumea staţiilor de lucru.

Liderul staţiilor de lucru inginereşti, SUN, şi-a întinerit toată gama de produse introducînd în 1989 zece noi modele pe linia maşinilor cu microprocesor 68 000 precum şi pe cele cu arhitectură cu set redus de instrucţiuni (RISC). Iată două din produsele lansate împreună cu caracteristicile lor:

- SUN D3/80 cu microprocesor 68030 la 20 MHz, memorie internă de 4 Mo şi sistem de operare UNIX sau MS/DOS. Performanţe: 3 MIPS;
- SUN D3/400 cu microprocesor 68030 la 33 MHz, memorie internă de 8-12 Mo. Performanţe: 7 MIPS. De remarcat că preţul este comparabil cu cel al unui calculator personal Macintosh II sau PS/2 Model 70.

De la aplicaţii inginereşti la birotică

Deşi în mod tradiţional staţiile de lucru au fost utilizate mai mult în aplicaţii inginereşti, un mare număr de staţii sînt utilizate în mod curent în alte domenii: industria prelucrătoare foloseşte staţiile pentru controlul proceselor, analiza calităţii, gestiunea stocurilor etc.

În aceste cazuri posibilităţile de multitasking şi interfeţele grafice sînt cerute de controlul în timp real. Domeniul publicaţiilor, în particular influenţat puternic de revoluţia electronică, este un consumator tot mai mare de staţii de lucru. Cum preţul staţiilor tinde să scadă mult, ele vor înlocui calculatoarele personale ca resurse desk-top pentru cei care au posibilităţi financiare mai mari. De exemplu, tra-

● Flash

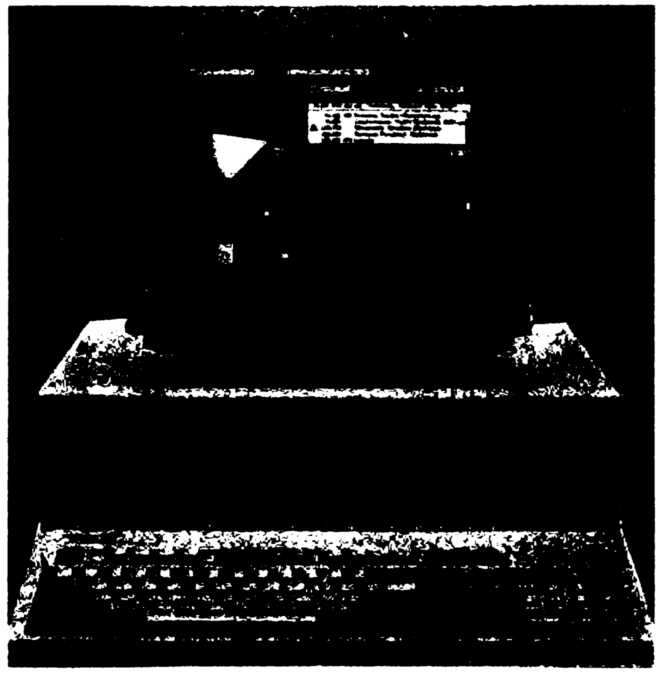
diţionala piaţă a automatizării birourilor (biroticii) dominată acum de PC-uri va fi caracterizaă în curînd de echipamente orientate pe interfeţe iconice, cu meniuri controlate şi ferestre etc., oferind astfel o productivitate de grup mai bună şi mai puţine posibilităţi de erori. Pentru întreprinderile medii, puterea de reţea a staţiilor poate rezolva problemele de achiziţionare a informaţiilor. Pentru manageri staţiile pot reprezenta o adevărată revoluţie, în sensul utilizării unor aplicaţii ca: **sisteme de raportare „on line“** care nu necesită probleme deosebite de aptitudini şi experienţă în utilizarea tastaturii.

O altă aplicaţie însemnată a staţiilor o reprezintă aşa-numita **„vizualizare a software-ului“.** Există o relativ nouă direcţie de dezvoltare, provenită din prezentări grafice, deosebit de utilă în cercetarea ştiinţifică fundamentală pentru realizarea de modele în domenii ca: dinamica fluidelor, prognoze meteorologice, fizica nucleară etc. Vizualizarea software este utilizată pentru a memora şi prelucra volume mari de informaţii produse de sisteme de prelucrare a imaginii şi de supercalculatoare. Staţiile de lucru prezintă un prilej unic, datorat posibilităţilor grafice şi de cuplare în reţele, în vederea controlului în timp real a acestor resurse.

Staţii de lucru sau... calculatoare tradiţionale?

Răspunsul la această întrebare este aproape imediat (sau chiar evident), deoarece diferenţa principală se referă la tipul interfeţei grafice utilizator. Pentru calculatoarele personale compatibile IBM precum şi tradiţionalele sisteme de prelucrare pe loturi utilizează interfeţe utilizator pentru linii de comandă similare. Pe aceste sisteme grafica este dedicată unor aplicaţii specifice ca, de exemplu, cele de prezentări grafice. Staţiile de lucru oferă un avantaj major faţă de prelucrarea convenţională (de tip personal sau pe loturi) existînd totuşi două obstacole care împiedică (încă) răspîndirea lor pe scară largă şi anume: costul şi lipsa uneltelor de dezvoltare de aplicaţii. Deşi costul staţiilor a scăzut simţitor ele sînt încă scumpe faţă de calculatoarele personale şi în special faţă de terminale.

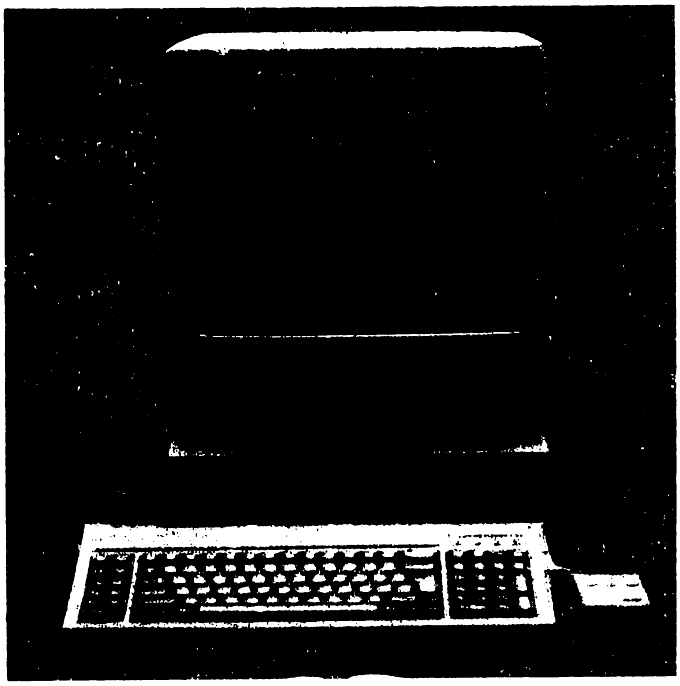
Cea mai importantă posibilitate de scădere a preţurilor poate veni însă din ideea de a privi staţiile ca **„terminalele super inteligente“** şi nu drept calculatoare din partea de jos a gamei. Dacă producătorii vor începe să ofere terminale cu procesoare adecvate, memorie suficientă, rezoluţie grafică înaltă precum şi posibilităţi de cuplare la reţele, atunci acestea vor putea realiza aplicaţiile specifice staţiilor de lucru.



COMPOQ DESKPRO 386/20TM

O astfel de staţie poate fi realizată pe o singură placă aşa cum sînt deseori proiectate terminalele. În acest caz, costul va fi (virtual) similar cu al unui terminal, diferenţa fiind legată de mărimea memoriei (care în prezent este ieftină) şi de microprocesorul utilizat.

„Uneltele“ grafice folosite în prezent sînt subrutine utilizate de programatori în urmă cu cel puţin 15 ani pentru dezvoltare de aplicaţii pe sisteme mari de calcul (mainframes) şi pe minicalculatoare. Mai recent, standardele CORE şi GKS au permis aplicaţii pe loturi cu care s-a utilizat puterea locală în grafică a terminalelor. După apariţia şi dezvoltarea calculatoarelor personale performante, acestea nu au integrat display-uri cu grafică interactivă şi cu facilităţi grafice deosebite (de exemplu, operaţii raster). Acest fapt precum şi puterea de prelucrare şi capacitatea de memorie prea mici pentru a realiza transformări de vectori au făcut ca PC-urile să nu poată fi utilizate în aplicaţii grafice de anvergură. Staţiile de lucru, însă, pot oferi aceste facilităţi, iar producătorii caută un model pentru standard.



SPARC STATION I

Din perspectiva stațiilor de lucru inginerești: calculatoare RISC

Calculatoarele cu set redus de instrucțiuni (RISC) au apărut pe piața stațiilor de 32 de biți, reușind în mai puțin de 3 ani să realizeze o adevărată invazie pe această piață (vezi „Știință și tehnică" nr. 3/1989). Toate calculatoarele RISC de astăzi exprimă ideile încorporate în trei sisteme care au reprezentat un pionierat la vremea loc. Este vorba de sistemele realizate de echipele de cercetători de la IBM, Stanford și Berkeley și care au văzut lumina zilei în perioada anilor 1980-1983. Cercetările efectuate la acea vreme au vizat obținerea unei viteze de prelucrare superioare la un preț redus. Primele sisteme realizare cu aceste concepte imitau, de fapt, primele calculatoare ale anilor '50. Acestea puneau la dispoziție puține instrucțiuni simple care funcționau cu date din memorie sau aduse în memorie. Nașterea tranzistorului și apoi a circuitelor integrate a permis realizarea în anii '60 și '70 a calculatoarelor cu circuite foarte rapide și unități centrale capabile să stocheze sute de instrucțiuni. Dar unitățile centrale realizate cu tehnologie pe siliciu erau foarte scumpe iar din acest motiv proiectanții vizau alte materiale pentru memoria de lucru. Astfel au apărut feritele, memorii mai ieftine (dar și mai lente) folosite pentru memorarea datelor.

Cercetările au vizat, apoi, creșterea vitezei prin reducerea drastică a numărului de transferuri de date extrase din memoria centrală, cu alte cuvinte prin scrierea unor instrucțiuni mai complexe. Au apărut astfel **calculatoarele cu set complex de instrucțiuni (CISC).** Pentru a nu încărca excesiv unitatea centrală s-a procedat la realizarea unui microcod (memorie de control) situat lîngă memoria centrală care să înglobeze instrucțiunile complexe. A rezultat o memorie centrală mai mică, însă operațiile erau mai lente deoarece fiecare instrucțiune complexă era decodificată atît de unitatea centrală cît și de memoria de control.

La sfîrșitul anilor '70 situația începuse să se modifice. Memoriile se ieftiniseră iar utilizarea memoriilor cache de viteză foarte mare a avut ca rezultat faptul că vitezele memoriilor au început să depășească vitezele memoriilor centrale.

În acest timp, prima cercetare realizată asupra modului în care programele erau executate de calculatoarele CISC (cercetare pornită în scopul sprijinirii dezvoltării de compilatoare) a dezvăluit un rezultat șocant: **numai 20% din setul de instrucțiuni îndeplineau 80% din operații.** Aceasta însemna că multe din instrucțiunile complexe erau, de fapt, redundante. Soluția prevăzută de IBM, Berkeley și Stanford a fost reîntoarcerea la arhitecturile construite cu micile inele de ferită.

Astăzi arhitecturile RISC au instrucțiunile implementate în partea de hard iar orice operație complexă este îndeplinită prin intermediul unui șir de instrucțiuni simple (mai lente). Se utilizează, de obicei, pînă la 32 de registre de uz general precum și memorii rapide (de tip cache) de dimensiuni mari. Viteza lor este îmbunătățită prin instrucțiuni de tip conductă (cu format fix, lungime de 32 de biți și puține moduri de adresare). Deși acest mic procent de instrucțiuni complexe este prelucrat mai lent, rezultatul final este reprezentat de o importantă creștere în viteza de prelucrare. Specialiștii în arhitecturile cu set redus de instrucțiuni prevăd că **viteza acestora va continua să se dubleze la fiecare 12-18 luni pentru următorii ani,** mai ales că se presupune și folosirea unor tehnologii performante ca: CMOS, ECL sau arseniură de galiu. Se observă astfel un avantaj net față de cel promis de microprocesoare ca Intel 80846 sau Motorola 68040. De aceea se prevede că arhitecturile RISC vor pătrunde în

circuitele P (Intel 80x86 şi Motorola 680x0). Deja IBM a introdus această tehnologie în staţia 6150 şi PC RT precum şi la unele din minicalculatoarele pe care le produce. Digital Equipment lucrează cu tenologiile RISC „în spatele uşilor închise". Interesul arătat de aceste uriaşe firme este datorat faptului că în prezent calculatoarele cu arhitectură **RISC oferă o viteză de 100 de ori mai mare faţă de VAX 11/780.**

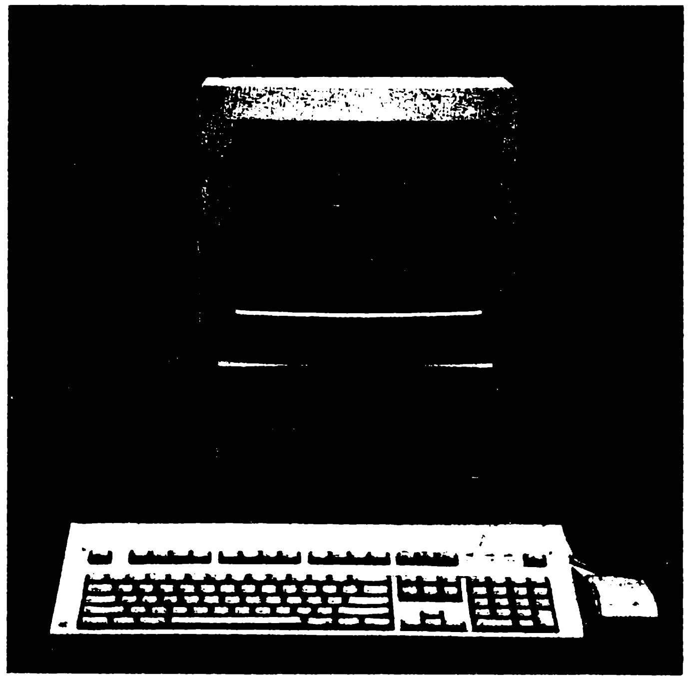
Sigur, există şi probleme. Cu cît sînt mai multe firme care comercializează echipamente cu arhitectură RISC încercînd să facă din ea un standard, cu atît mai grea va fi selectarea unei opţiuni.

Cîntărind factorii RISC

Din punct de vedere al modului în care sînt realizate, arhitecturile RISC sînt, în acelaşi timp, atît diverse cît şi complexe. De exemplu, echipele de la IBM şi Stanford împing o serie de procese spre software în scopul optimizării compilatoarelor. În alte locuri (Berkeley) se preferă implementarea mai multor operaţii complexe în hardware prin maximizarea registrelor unităţii centrale. Astăzi proiectanţii de arhitecturi RISC au de ales între cîteva alternative şi anume: IBM 801 (care prezintă 120 de instrucţiuni), Stanford RISC (139 de instrucţiuni) şi Berkeley MIPS (55 de instrucţiuni). Se observă că nu există o mărime standard pentru setul de instrucţiuni. Mai mult, unele maşini realizează operaţii în uirgulă mobilă prin software, în timp ce altele le inserează în partea de hard sau folosesc coprocesoare de virgulă mobilă.

Toate aceste diferenţe trebuie să fie, însă, invizibile utilizatorilor finali ai staţiilor de lucru, aplicaţile UNIX standard putînd fi modificate pentru a rula pe orice maşină prin compilatoare standard. În schimb, un circuit „standard" este mult mai dificil de impus. Se pare, însă că Sun este pe cale să realizeze acest lucru cu produsul său SPARC (Scalable Processor Architecture). SPARC este „fiul" lui RISC (Berkeley), rivalii săi fiind Precision Arhitecture (HP), Pyramid precum şi unele produse IBM.

Sun şi-a conceput arhitectura astfel încît, dacă la început setul redus de instrucţiuni era identificabil în special pe staţiile de lucru Sun, în prezent apare din ce în ce mai frecvent la calculatoare personale, eclipsînd tehnologia CISC prezentă, în special, la microprocesoarele Intel. Setul redus de instrucţiuni prezintă mai puţine instrucţiuni încorporate în hardware avînd o schemă mult mai eficientă. Din aceste motive este mai rapid faţă de CISC însă mai costisitor.

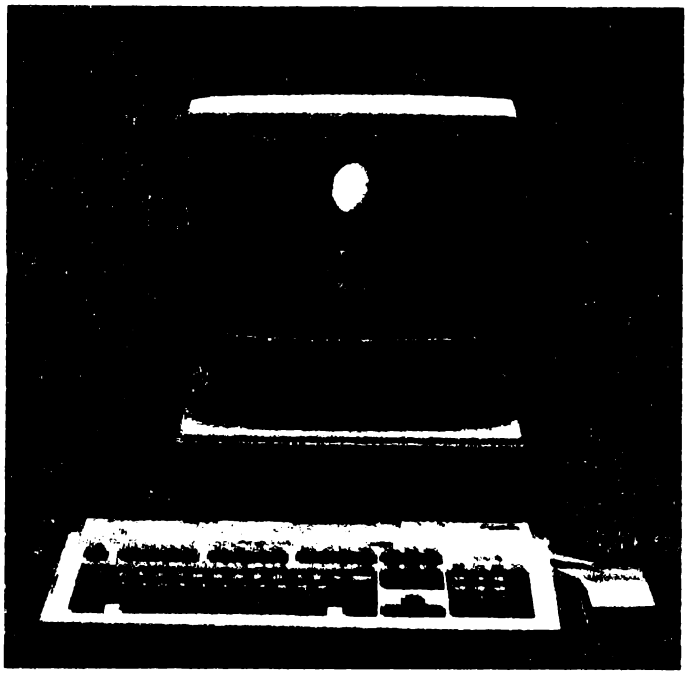


APPLE MACHINTOSH CX

SPARC prezintă o unitate centrală care constă într-o unitate de calcul cu numere întregi (Integer Unit) şi o unitate de calcul în virgulă mobilă concurentă (Floating Point Unit) care poate fi implementată în acelaşi circuit. Are la bază tehnologia RISC (numai 50 de instrucţiuni) şi optimizează calculele în virgulă flotantă în comparaţie cu rivalii săi IBM şi MIPS. A fost proiectat în mod special pentru rularea programelor în limbaj C şi sub sistem de operare UNIX.

De remarcat faptul că deja au început să apară copii (imitaţii) după arhitecturile SPARC („clones"-uri) care prezintă preţuri de 10% mai mici faţă de cele originale. De asemenea, producătorii de circuite RISC au început să se alinieze la standardul SPARC, existînd deja, 5 firme importante (Texas Instruments, Fujitsu, LSI Logic, Bipolar Integrated Technology şi Cypress) care oferă seturi de circuite compatibile SPARC. Ultimul set de circuite compatibile SPARC oferit de firma Cypress prezintă ca performanţe 24 MIPS faţă de 6 MIPS comparativ cu microprocesorul Intel 80386 la 25 MHz.

Copiile după arhitecturile SPARC rulează sub sistem de operare UNIX fiind puse la dispoziţie şi circa 400 de programe de aplicaţii. Se prognozează că în anul 1990 preţul acestor staţii de lucru va fi cuprins între 5 000 şi 10 000 de dolari.



ARCHIMEDES 410

Iată cîteva tipuri de cele mai cunoscute stații de lucru SPARC împreună cu cele mai importante caracteristici tehnice ale lor:

● **SPARC Station I** prezintă o viteză de 12,5 MIPS (1,4 MFlops) iar pentru îmbunătățirea acesteia se poate atașa un coprocesor în virgulă mobilă. Capacitatea de memorie internă este de 8-16 Mo, iar a celei de masă de 160, existînd și unitate de bandă backup. Are în componență un procesor (accelerator) pentru prelucrarea imaginilor și sînt puse la dispoziție circa 500 de pachete de programe (Sparcware).

● **SPARC Station 300** prezintă o viteză de 16 MIPS (2-6 MFlops), memorie internă de 8-40 Mo iar memoria de masă 1,3 GO.

● **SPARC Station 370** are o capacitate de memorie internă și de masă sporite (56 Mo și respectiv 5,6 GO).

Iată și caracteristicile tehnice ale unei stații de lucru cu circuit RISC realizată de firma Tektronix: D88 realizată cu microprocesor Motorola 88 000; viteza: 14-17 MIPS (7-12 MFlops); memorie internă: 8 Mo.

Cu toate că arhitecturile SPARC par să devină un standard și alte firme dezvoltă arhitecturi noi pornind de la cele cu set redus de instrucțiuni. Motorola, HP și DEC realizează arhitecturi RISC optimizate pentru rularea sub UNIX. Motorola 88 000, de exemplu, prezintă 51 de instrucțiuni și 32 de registre, iar unitățile de calcul în numere întregi și în virgulă mobilă sînt conținute într-un singur circuit. Una din cele două unități de memorie rapidă (cache) standard este folosită pentru date iar celelalte pentru instrucțiuni. Pot fi incluse pînă la 8 unități de memorie cache astfel încît arhitectura este proiectată pentru diverse tipuri de aplicații.

Cum lista implementărilor RISC crește este de presupus că se va mări și cea a diferențelor între caracteristici. Iată și alte realizări: **2 900 (AMD), Precision Arhitectura (HP), ABAQ (Atari), Acorn, RISC Machine (Acorn, Marea Britanie).** Cel din urmă exemplu pare a fi optim pentru piața calculatoarelor personale. Prin utilizarea unei mici suprafețe de siliciu și a cîtorva tranzistoare se preconizează o reducere importantă a prețului.

Cu aceste implementări este improbabil faptul că un singur ofertant RISC va putea să se impună singur ca un standard de factor în viitorul apropiat, mai ales în condițiile în care programele pentru utilizatori vor rămîne portabile la un nivel ridicat. Astfel ca și în alte cazuri, testul pentru arhitecturile RISC va fi reprezentat de capacitatea lor de a realiza aplicații specifice.

MAjoritatea sistemelor RISC depind de compilatoarele lor, iar performanțele, de gradul de optimizare față de un anumit limbaj sau sistem de operare.Multe din aceste stații de lucru sînt realizate cu materiale și tehnologii avansate (de exemplu, versiunea cu arseniură de galiu) care le adaugă performanțe suplimentare. Dar se așteaptă ca sistemele cu set redus de instrucțiuni să producă cele mai mari performanțe atunci cînd setul redus de instrucțiuni va fi combinat cu prelucrarea paralelă care va pune probabil în frunte hibridul RISC **Transputerul firmei Inmos** și arhitectura **PRISM** (Apollo). Arhitectura PRISM (Paralel Reduced Instruction Set Multiprocessing) prezintă circuite de 64 de biți și ajunge la o viteză de peste 100 de ori superioară lui VAX 11/780.

Piața de stații de lucru prezintă o rată de creștere de două ori mai mare de cea a întregii industrii de calculatoare. Primii doi mari în domeniul tehnicii de calcul și informaticii mondiale, IBM și DEC, recunosc importanța strategică a acestor calculatoare! În acest context este posibilă orice răsturnare de situație în care să cîștige piața o firmă ce este astăzi poate necunoscută, deoarece în lumea informaticii ultimului deceniu succesul poate veni uneori foarte repede! Dar și reversul lui...


actualitatea pc
SUPPORT QUALITY
1
1
1

Doina ISTRĂTESCU
Eugen GEORGESCU

Cuploarele grafice ale calculatoarelor compatibile IBM PC

Unul dintre elementele ce au contribuit în mare măsură la explozia pe piaţă a calculatoarelor personale a fost grafica, deoarece reprezintă un mod de comunicare sintetic, mai adaptat simţurilor umane şi mai flexibil. O serie de noutăţi pe plan tehnologic şi conceptual au făcut ca unele modele de vîrf din clasa calculatoarelor personale să fie tolerate în clubul aristocratic al echipamentelor grafice profesionale, deşi preţul lor de cost este mult mai scăzut. În literatura de specialitate această stare de fapt este caracterizată prin sloganul „high-end PC²low-end Workstation", **lansat evident de producătorii de PC-uri.**

Pînă la apariţia PC-urilor, a face grafică însemna în general o prelucrare de date pe o unitate centrală şi o transmisie a datelor către un terminal folosind un protocol serial asincron sau sincron, la viteze cuprinse între 2 400 bps şi 2 Mbps. Acest terminal a evoluat foarte mult ca posibilităţi, preluînd la nivel local aproape tot ce înseamnă grafică: trasări de elemente geometrice, umplerea (filling) contururilor cu diverse modele (pattern) prezentarea în spaţiu tridimensional. Cu această ocazie s-a schimbat şi denumirea din terminal grafic în staţie grafică terminală, unul din cele mai populare modele fiind IBM 5080. Evoluţia nu s-a oprit însă aici, staţiile grafice moderne fiind actualmente unităţi de calcul autonome care înglobează şi unul sau mai multe procesoare pentru deservirea activităţilor de grafică.

Calculatoarele personale nu au mers însă pe această linie costisitoare, ele speculînd în mod inteligent o caracteristică proprie şi anume aceea că memoria video este conţinută în spaţiul propriu de adresare al. procesorului. Imaginea este formată în memoria video direct fără a mai fi nevoie de transmisie pe linie serială, obţinîndu-se astfel o viteză de lucru foarte mare. Primele care au beneficiat de aceasta au fost jocurile şi în general programele care necesitau animaţie. Scrierea direct în memoria ecran reprezintă numai o fază a procesorului. Pentru afişarea propriu-zisă această memorie video este citită secvenţial de un bloc logic, independent ca funcţionare de procesor, care realizează semnalele pentru monitorul TV conform standardului acestuia. Deşi această funcţie pentru cuplorul grafic nu este nici singura şi nici cea mai complicată, ea este cea mai importantă şi cu prioritate maximă la accesul resurselor.

Calculatoarele pentru care sînt valabile principiile mai sus enunţate sînt foarte numeroase, dar ne vom rezuma în cele ce urmează la prezentarea cuploarelor grafice (graphics adapter) cele mai utilizate în calculatoarele compatibile **IBM PC/XT/AT** şi în noile modele **PS/2.**

Din punct de vedere istoric primele cuploare grafice ce au echipat calculatoarele din gama IBM PC/XT au fost **Hercules** şi **CGA.** Cuplorul Hercules este de tip monocrom dar de bună rezoluţie. Preţul de cost relativ redus al ansamblului cuplor-monitor i-a adus o oarecare popularitate la începutul anilor '80. Particularităţile sale constructive nu l-au făcut apt pentru dezvoltări ulterioare şi practic a fost eliminat de cuplorul **CGA (Color Graphics Adapter)** care, deşi oferea o rezoluţie mai redusă, (320x200) aducea o nouă dimensiune, şi anume culoarea. Acest cuplor a devenit practic un standard pentru calculatoarele XT deşi poate fi întîlnit şi în unele configuraţii ieftine de AT. Răspîndirea a fost impulsionată într-o măsură decisivă de avalanşa de software grafic pentru domenii ca **proiectarea asistată, statistici economice** şi bineînţeles **jocuri.** Totuşi

acesta a fost numai un început fiindcă CGA prezenta o serie de inconveniente majore cum ar fi: rezoluţia redusă; număr de culori simultan afişabile foarte redus şi gama de culori — din care se putea face selecţia — aproape nesemnificativă.

La fel ca şi la sistemele de televiziune, următoarele cuploare grafice au urmat calea firească a compatibilităţii: deşi au apărut moduri noi de lucru, cuploarele răspundeau şi la vechile pachete software. Se remarcă astfel în 1984 apariţia cuplorului de tip **EGA (Enhanced Graphics Adapter)** care prin concepţie se apropie de echipamentele grafice profesionale, deşi performanţele continuă să rămînă deficitare la unele capitole cum ar fi numărul de culori afişabile simultan, paleta din care pot fi acestea selectate şi într-o oarecare măsură chiar rezoluţia. Totuşi această apariţie este un eveniment deoarece introduce o serie de elemente funcţionale ca planele de culoare, tabela de culori (Color Table) şi ALU (Arithmetic Logic Unit) pentru scrierea pixelilor, elemente ce sînt specifice echipamentelor grafice performante.

În anul 1987, firma IBM introduce o versiune perfecţionată denumită **VGA (Video Graphics Array)** care devine echipare standard pentru noua serie PS/2 la modelele 50, 60 şi 80 unde este implementată direct (built-in) pe placa de bază (mother-board). Noul cuplor păstrează compatibilitatea cu modelele anterioare dar aduce un spor de rezoluţie, creşte fantastic paleta de culori şi într-unul din modurile de lucru, poate afişa 256 de culori simultan dar din păcate la rezoluţie redusă (320x200). Acest mod era impus de un nou domeniu de activitate şi anume prelucrarea de imagini.

Deşi nu se incadrează strict în tematica acestui aricol vom încerca să arătăm foarte pe scurt de ce la ora actuală numărul de culori simultane şi paleta din care se pot selecta tind să devină mai importante decît rezoluţia. Pentru a afişa cît mai veridic un obiect tridimensional pe suprafaţa bidimensională a unui ecran e necesară să dăm impresia de relief prin umbriri, deci prin gradaţii de culoare şi saturaţie. O imagine TV de bună calitate acceptă aproximativ 5 000 de gradaţii. La un cuplor de tip VGA cele 256 de culori simultane, pot fi de fapt 256 gradaţii de luminanţă ale aceleiaşi culori. Această problemă mai capătă o dimensiune dacă vrem să redăm şi suprafaţa obiectului (lucioască, mată, raster etc.) care înseamnă un sistem de microreflexii, deci de microgradaţii de culoare. Pentru a încheia prezentarea cuploarelor grafice de largă răspîndire ar mai trebui remarcat că pentru PS/2 Model 30 firma IBM a realizat cuplorul MCGA (Multi-Color Graphics-Array) care este compatibil cu CGA.

În continuare vom prezenta în sinteză principalele facilităţi ale cuploarelor menţionate.

În cadrul tabelului 1 apariţia de mai multe ori a unui cuplor înseamnă de fapt moduri diferite de lucru ale acestuia.

În această expunere au fost complet neglijate modurile de lucru alfanumerice. Şi aici posibilităţile au evoluat foarte mult de la CGA la VGA. Astfel, numărul de linii de text afişabile pe un ecran a crescut de la 25 (CGA), 43 (EGA) pînă la 50 (VGA). Matricea de definire a unui caracter de 8x14 pixeli (CGA) a căpătat noi versiuni: 8x8 (EGA)şi 8x16 (VGA). Se pot utiliza concomitent pînă la 4 tipuri de seturi de caractere (EGA/VGA) care pot fi definite şi utilizate on-line. Această ultimă facilitate este importantă cînd se doreşte utilizarea simultană a mai multor seturi de caractere (latin, chirilic, gre-

TABELUL 1

CUPLOR	REZOLUŢIE	NR. DE CULORI SIMULTANE	PALETA	MEMORIA VIDEO (kB)	BIOS
Hercules	72Øx348	2	2*	4	System
CGA	32Øx2ØØ	4	2**	11	System
CGA	64Øx2ØØ	2	2*	16	System
EGA	64Øx35Ø	16	64	256	Extins
VGA	64Øx48Ø	16	256 K	256	Extins
VGA	32Øx2ØØ	256	256 K	256	Extins
VGA***	8ØØx6ØØ	16	256 K	256	Extins

* – *Mod monocrom*

** – *Există doar două tipuri de combinaţii de culoristandard, denumite paletele CMW (Cyam, Magenta, White) şi GRY (Green, Red, Ywllow). La unele jocuri s-a dovedit că există şi o a treia paletă nestandard.*

*** – *Acest mod nu este standard şi poate fi implementat prin atacul direct la posturi al cuplorului cu condiţia ca monitorul să aibă o frecvenţă de scanare orizontală de cel puţin 40 kHz (ex.: NEC Multisync II).*

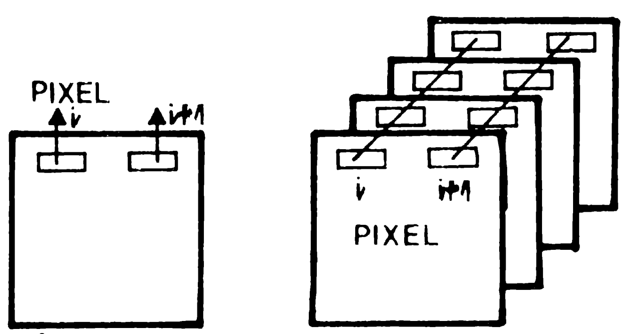
cesc, arab) sau a unor seturi de caractere naționale (ex.: ă, î, ş, ţ în română). Descrierea acestor facilități va face obiectul unui alt articol.

Exploatarea posibilităților unui cuplor se poate face în general prin intermediul BIOS-ului şi anume cu ajutorul întreruperii 10H. Pentru cuploarele EGA şi VGA, BIOS-ul capătă o extensie care nu se mai găseşte pe placa de bază ci în nişte PROM-uri situate fizic chiar pe cuplor. Pentru aplicații performante, în care se doreşte exploatarea la maxim a tuturor facilităților unui cuplor, iar factorul timp este prohibitiv, se recomandă folosirea directă a porturilor.

Pentru înțelegerea unor principii funcționale moderne şi ținînd cont de tendințele actuale, accentul prezentării se va pune pe EGA şi VGA, referirile la CGA fiind sumare.

MEMORIA VIDEO

Este unul din elementele funcționale de bază în cadrul oricărui sistem grafic. Performanțele ei (timpul de acces şi dimensiunea) influențează decisiv rezoluția iar forma de organizare are efect asupra numărului de culori afişabile simultan. Există două forme principale de organizare.



a. ÎMPACHETATĂ b. PLANE DE CULOARE

FIGURA 1

Prima versiune foloseşte o organizare bidimensională a memoriei şi din punct de vedere istoric are prioritatea apariției. Este folosită la CGA. Să vedem cazurile concrete cum se realizează aceasta.

CGA 320x200/4 culori

Folosindu-se numai 4 culori simultane, pentru codificarea culorii unui pixel sînt necesari numai 2 biți, deci în cadrul unui octet putem avea 4 pixeli, deci pentru o linie avem nevoie de 80 de octeți. Pentru acest mod adresa de început a memoriei video este B800:0000H (segment:offset). Astfel dacă vom scrie la această adresă octetul:

0	0	0	1	1	0	1	1	
0		1		2		3		PIXEL

vom observa în colțul din stînga sus un pixel de culoarea fondului (background) şi alți 3 pixeli de culori diferite (foreground). De remarcact că la CGA între culoarea care se observă fizic pe ecran şi alegerea codificării nu există decît două variante aşa cum s-a mai arătat.

CGA 640x200/2 culori

Acesta este un mod de lucru monocrom care duce însă la dublarea rezoluției pe orizontală. În acest mod pentru fiecare pixel este necesar un singur bit. Adresa de început a memoriei video în acest caz este tot B800:0000H. Avînd în vedere rezoluția se observă că pentru o linie sînt necesari tot 80 de octeți. Apare totuşi o ciudățenie în organizarea memoriei video şi anume: liniile impare sînt grupate în spațiul de adrese începînd de la B800:0000H iar liniile pare sînt grupate începînd de la BA00:0000H. La un calcul simplu se poate deduce că între cele două zone rămîne un spațiu de 192 de octeți nefolosiți. Pentru a înțelege acest mod se poate face o experiență simplă şi anume, dacă vom scrie cîte 80x100 octeți de la adresele mai sus menționate cu conținutul:

1	0	1	0	1	0	1	0	0AAH

vom observa că ecranul s-a umplut cu nişte linii verticale foarte fine. Dacă în acest loc vom folosi alt octet:

0	0	0	0	1	1	1	1	0FH

vom observa nişte bare verticale groase.

Aceste experiențe se pot face direct din DEBUG-er cu condiția ca înainte de a accesa memoria video să trecem în modul 4 (320x200) sau 6 (640x200) folosind întreruperea 10H din BIOS.

A doua versiune de organizare a memoriei video, cea din plane de culoare, este incomparabil mai complexă dar este singura capabilă să asigure un nivel înalt al performanțelor. Dacă în

versiunea anterioară vom încerca să mărim numărul de culori sau rezoluţia vom avea dificultăţi cu timpul de acces, fiindcă viteza de baleiaj a monitorului TV trebuie să rămînă între anumite limite. Deci dacă la citirea octet cu octet nu putem depăşi o anumită viteză dictată de factori tehnologici, atunci o soluţie ar fi să citim în paralel mai multe memorii dar la o viteză mai redusă.

Această structură are însă o complexitate sporită prin ataşarea obligatorie a altor componente funcţionale. Pentru exemplificare prezentăm în tabelul 2 o versiune simplificată a cuplorului EGA. Diferenţele faţă de VGA sînt relativ mici şi în general o astfel de structură seamănă principial cu echipamentele grafice profesionale.

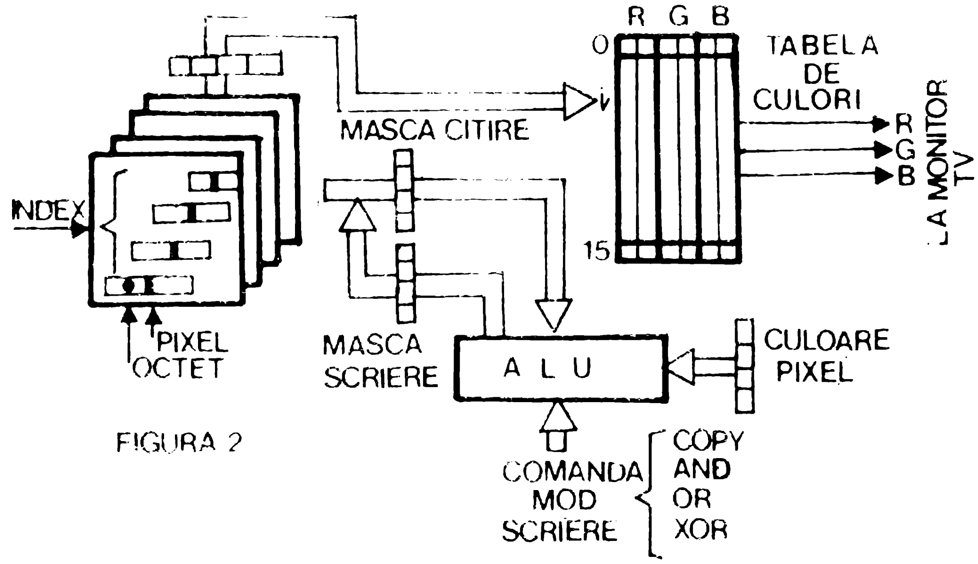


FIGURA 2

Vom începe descrierea structurii pornind de la planele de culoare. Dacă luăm cazul unui singur plan de culoare vom observa că în interiorul fiecărui octet, fiecărui bit îi corespunde un pixel, la fel ca la funcţionarea monocromă studiată anterior. Deci un plan de culoare implementează practic aspectul bidimensional x-y al monitorului TV. Pentru display (afişaj) culoarea apare ca o a treia dimensiune iar aceasta se transpune în memoria video ca axa z, deci adîncimea. Vom explica acest lucru mai bine printr-un exemplu. Adresa de început a memoriei video în acest caz este A000:0000H, iar toate cele 4 planuri răspund fizic la aceeaşi adresă. Modul în care se rezolvă acest conflict în operaţii de scriere/citire va fi explicat în cursul prezentării cuplorului. Dacă la adresa de început avem următorii octeţi:

0	1	0	1	0	0	1	1	PLAN 0
0	0	1	1	0	0	0	1	PLAN 1
0	0	0	0	1	0	1	1	PLAN 2
0	0	0	0	0	1	0	1	PLAN 3

atunci în colţul din stînga sus al ecranului vom avea o succesiune de pixeli de următoarele culori:

PIXEL	CULOAREA
1	0
2	1
3	2
4	3
5	4
6	8
7	5
8	15

După cum se observă codurile de culoare ale pixelilor se obţin prin compunerea pe adîncime (axa z) a pixelilor corespunzători.

Acest cod nu are un corespondent fix în spaţiul culorilor afişabile care, aşa cum am văzut, poartă denumirea de paletă. Codul este numai un fel de „culoare logică" adică un index într-o tabelă de culori aşa cum se observă în tabelul 2. La locaţia selectată în acest tabel cu ajutorul indexului se găsesc acele informaţii care dau efectiv culoarea pe ecran. Deci Tabela de Culori face translaţia de la „culoare logică" la „culoare fizică". Această modali-

tate de transcodare provine din tehnicile de codare unde mai poartă denumirea de Look-Up Table, denumire care se găseşte ca alternativă în documentaţiile unor firme din S.U.A.

Pentru a înţelege mai bine rolul acestei tabele este necesar să facem o scurtă divagaţie pe teme de colorimetrie. Orice culoare existentă în natură, indiferent de nuanţă şi saturaţie poate fi obţinută şi ca o sumă a trei culori combinate în diverse proporţii. Cele trei culori nu pot fi alese oricum, ele trebuie să respecte anumite reguli şi poartă denumirea de culori primare. Această metodă poartă denumirea de sinteză aditivă şi setul de culori primare pentru monitoarele TV este roşu, verde, albastru sau după o formulă consacrată RGB (Red-Green-Blue).

În cazul cuplorului EGA, pentru fiecare culoare primară pot exista doar 4 intensităţi deoarece ele se codează în Tabela de Culori pe cîte 2 biţi. Aici se găseşte şi unul din punctele neplăcute ale lui EGA şi anume faptul că pentru o culoare nu putem obţine în cel mai bun caz decît 3 intensităţi, cea de-a patra fiind intensitate nulă, deci negru. Exemplu:

R		G		B		
0	0	0	0	0	0	NEGRU
0	1	0	1	0	0	GALBEN 25%
1	0	1	0	0	0	GALBEN 50%
1	1	1	1	0	0	GALBEN100%

Una din tehnicile de a mai obţine diverse luminozităţi este de a mai adăuga sau scădea, dacă este posibil, culoarea albă. Aceasta înseamnă a umbla la saturaţia culorii dar un ochi neexperimentat va accepta aceasta ca o variaţie de luminozitate. Acest artificiu ne va ajuta ca pentru o culoare primară R, G sau B să mai putem obţine 2 nuanţe, dar se pot găsi şi combinaţii la care să nu mai putem obţine altă luminozitate, ca de exemplu:

R		G		B	
1	1	1	0	0	1

Se observă că nu putem face operaţii ± fără a altera culoarea care, în acest caz, poate fi definită ca un fel de galben ce bate spre violet.

Privind din nou la tabela de culori se observă că toate combinaţiile ce se pot realiza cu cele trei culori fundamentale sînt în număr de 64(2^6). Caracterizînd cuplorul EGA prin rezoluţie, numărul de culori simultane şi paletă, se poate afirma că aceşti parametri se degradează faţă de minimul cerut unei aplicaţii profesionale tot în această ordine, ultimul parametru, paleta, fiind de-a dreptul dezamăgitor pentru că numărul de biţi pentru o culoare primară e mai mic decît numărul de plane.

În concluzie, poziţia fizică a unui pixel pe ecran este dată de adresa bitului în cadrul planului de culoare, iar culoarea sa are o valoare codificată prin combinaţia biţilor corespunzători din toate planurile. Această valoare codificată primeşte un corespondent fizic în paleta de culori printr-o transformare la sistemul RGB utilizînd Tabela de Culori.

Din punct de vedere constructiv această Tabelă de Culori este o altă memorie, cu un timp de acces foarte mic. Numărul de adrese al acestei memorii speciale este în strînsă legătură cu numărul de planuri de culoare şi este egal cu numărul de culori afişabile simultan. (Numărul de culori este egal cu doi la o putere egală cu numărul de planuri). Sistemele grafice performante lucrează de obicei cu 8 sau 12 planuri, adică pot afişa 256 şi respectiv 4 096 de culori.

Dimensiunea cuvîntului în Tabela de Culori este, de asemenea, deosebit de importantă pentru cazurile în care vrem să facem umbriri sau să modelăm textura suprafeţei. Deşi luate independent, ochiul nu poate distinge foarte multe nuanţe, totuşi în imagini de ansamblu sînt absolut necesare. Încercaţi să ghiciţi cîte nuanţe sînt în reflexul unei raze de soare pe suprafaţa unui pahar cu apă. Dacă soarele este puternic sînt peste 10 000. Pentru astfel de modelări la unele sisteme, cum ar fi Hewlett-Packard 9 000/seria 300, Tabela de Culori are cîte 8 biţi pentru fiecare componentă RGB.

După cum se observă însă din tabelul 2, indexul de culoare al unui pixel nu atacă direct Tabela de Culori, ci trece printr-o Mască de Afişare (Display Enable Mask) care poate face ca unele planuri să nu participe la formarea indexului de culoare, ele fiind considerate 0. În aceste condiţii în acel plan se poate face orice operaţie de desenare fără ca în acest timp să se altereze imaginea de pe display.

Prezentarea unor astfel de detalii interne din structura unor cuploare grafice pare lipsită de sens în condiţiile în care compilatoarele actuale au biblioteci grafice atît de uşor de utilizat. Inconvenientul major este că acestea

doresc să acopere o gamă cît mai largă de cuploare şi, în acest caz, ele se bazează numai pe posibilităţile comune. De exemplu, bibliotecile compilatoarelor produse atît de Microsoft cît şi de Borland nu oferă nici o facilitate de a exploata Masca de Afişare care, împreună cu Tabela de Culori, stau la baza animaţiei pe calculator, atît de gustată dar greu de exeplicat pentru cei neavizaţi. Dacă veţi privi cu atenţie programele de animaţie, veţi observa că numărul de culori este redus, semn sigur că unul sau două planuri sînt în umplere cu noua imagine iar celelalte se afişează. Cu cît programul este mai „vioi" cu atît el are mai puţine culori simultane. Altă versiune este ca decorul să fie desenat în unul sau două planuri iar restul să fie folosite pentru elementele în mişcare. Cunoscînd acum aceste detalii priviţi cu atenţie şi veţi descoperi „secretele de fabricaţie".

Cuplorul de tip VGA are o serie de îmbunătăţiri. Prima este o creştere a rezoluţiei pe verticală cu circa 30%, preţul plătit pentru aceasta fiind renunţarea la a doua pagină. A doua îmbunătăţire este cu totul spectaculoasă şi constă în creşterea numărului de biţi pe componentă RGB de la 2 la 6 biţi, deci paleta a crescut de la 64 la 256 K culori! Tabela de Culori a fost menţinută din considerente de compatibilitate, dar ea nu mai conţine informaţia RGB, ea devine doar un nou codor care, prin combinaţii cu alte registre, formează un sistem de adresare a 255 de Registre de Culoare conţinînd informaţia RGB. Sistemul de adresare este destul de sofisticat şi diferit pentru modurile de lucru, aşa că nu vom intra în detalii. Tot ca o îmbunătăţire remarcabilă este şi modul de lucru cu 256 de culori simultane dar din păcate la o rezoluţie redusă (320x200). Organizarea memoriei are în acest caz un aspect hibrid şi anume ea este organizată pe 4 planuri dar în fiecare plan la codare participă cîte 2 biţi ca la CGA în modul cu aceeaşi rezoluţie. Acest lucru este însă transparent pentru utilizator pentru care există corespondent direct între pixel şi octet începînd de la adresa A000:0000H.

O ultimă îmbunătăţire la VGA faţă de EGA este şi faptul că registrele cuplorului pot fi citite, deci starea cuplorului poate fi determinată cu exactitate în orice moment.

Toate detaliile prezentate anterior s-au referit la citirea memoriei video în scopul formării imaginii pe monitorul TV. Această citire se face ciclic sub controlul unui bloc funcţional care poartă denumirea de Controller CRT. El asigură formarea semnalelor de culoare RGB şi sincro pentru monitorul TV.

Vom studia în continuare interacţiunea dintre procesor şi memoria video. În cazul în care memoria este de tip împachetat, ea este adresată normal de către CPU, dar în varianta în care se lucrează cu planuri de culoare, lucrurile sînt mult mai complicate deoarece planurile sînt mapate practic pe aceleaşi adrese. Ele ar putea fi teoretic tratate cu o magistrală de 32 de biţi, dar în general cuplorul EGA este conceput să poată funcţiona şi pe calculatoare IBM PC/XT care au magistrala de 8 biţi. În această situaţie există două metode de acces.

În prima metodă procesorul poate adresa fiecare plan în parte. Această metodă nu este folosită decît în aplicaţii speciale (ex.: animaţie) deoarece scrierea completă a unui pixel ar cuprinde şi 4 operaţii de selectare separată a fiecărui plan de culoare. În plus, deoarece indexul de culoare s-ar forma bit cu bit pe măsura completării planurilor, efectele ar fi incontrolabile şi neplăcute pentru ochi.

A doua metodă constă în scrierea simultană a tuturor planurilor. Acest lucru se realizează cu ajutorul unui Registru de Culoare Pixel. Reprogramarea acestui registru este necesară numai cînd se schimbă culoarea de trasare, deci mult mai rar. Din punct de vedere al procesorului, scrierea ar decurge ca într-un singur plan deşi se completează toate conform registrului.

Aceasta a fost totuşi o viziune simplificată. Dacă revenim la figura 2 vom observa, pe lîngă o serie de registre de mascare la scriere şi citire, un important element funcţional şi anume ALU (Arithmetic Logic Unit) care poate face ca noua culoare a unui pixel să fie dependentă de vechea valoare. În cuploarele EGA şi VGA circuitul ALU folosit poate executa patru funcţii: COPY; AND; OR; XOR.

Faţă de alte echipamente grafice performante, numărul de funcţii implementat e mic, dar sînt cele utilizate în 80% din aplicaţii. Cu titlu de curiozitate menţionăm că la calculatorul Felix M118 circuitul ALU are 16 funcţii!!! Funcţia COPY atunci cînd se face desenarea prin supraimprimare. Funcţiile AND şi OR sînt folosite pentru desenări cu efecte de transparenţă. Foarte importantă este funcţia XOR care face inversare şi reprezintă mecanismul de bază pentru cursoare grafice şi unele versiuni de animaţie.

Aşa cum s-a precizat anterior, una din metodele de a folosi resursele cuploarelor EGA/VGA este prin inter-

mediul BIOS-ului extins prezent pe placă. Această metodă este folosită de multe cuploare inteligente cum ar fi de exemplu cel de hard disk. Întreruperea 10H care deserveşte ecranul, practic se dublează ca număr de funcţii pentru operaţiuni de grafică, culori, scriere texte, seturi de caractere. Adresa la care începe BIOS-ul extins este C0000:0000H şi conţine în primii trei octeţi o semnătură. Dacă în cursul operaţiei de bootstrap se depistează această semnătură se lansează programul de la C000:0003H care face toate operaţiunile necesare integrării cu BIOS-ul de bază. Detalii concrete asupra funcţiilor de la întreruperea 10H se pot găsi în bibliografia de specialitate sau folosind programul HELP.

Pentru operaţii care nu sînt prevăzute în BIOS se impune utilizarea directă a registrelor cuplorului care sînt grupate în 5 seturi de bază. Adresele lor sînt prezentate în tabelul 2.

TABELUL 2

GRUP	ADRESE (HEX)
General	3DA, 3CA, 3C2, 3CC
Sequenser	3C4, 3C5
CRT Controller	3D4, 3D5
Graphics	3CE, 3DF
Atribute	3CØ, 3C1

În continuare vom descrie succint fiecare grupă şi modul de acces.

Registrele generale. Nu sînt interesante pentru creatorul de soft. Unele dintre ele sînt imaginea unor pini din conectorul monitorului iar altele sînt legate de configurarea cuplorului.

Sequenser. Conţine o serie de 5 registre ce controlează funcţii referitoare la modurile de lucru alfanumerice şi seturile de caractere. Toate registrele sînt cuplate pe o magistrală comună şi răspund la aceeaşi adresă. Selecţia lor se face prin intermediul unui demultiplexor ca în figura 3.

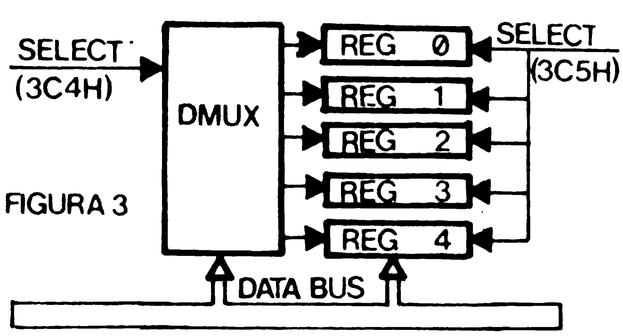


FIGURA 3

Pentru a putea accesa un registru la un moment dat, trebuie să efectuăm două secvenţe: selecţia în multiplexor şi scrierea datelor. Dacă se transmit succesiv multe date către acelaşi registru, selecţia multiplexorului e suficient să se facă o singură dată la începutul secvenţei.

Un exemplu în limbaj C ar fi:

```
outp (0x3C4, nr.registru);
outp (0x3C5, valoare).
```

Pentru creatorii de grafică, în acest grup este interesant numai registrul 2 care conţine o mască pentru selecţia planurilor de culoare în care este permisă scrierea.

CRT Controller. Este format dintr-o serie de 26 de registre care conţin în general toţi parametrii necesari pentru formarea semnalelor de comandă ale monitorului TV. Modul de adresare este similar cu cel de la sequenser, adresele demultiplexorului şi registrelor fiind 3D4H şi respectiv 3D5H. Schimbarea parametrilor funcţionali ai acestui bloc poate avea efecte spectaculoase dar şi unele catastrofice.

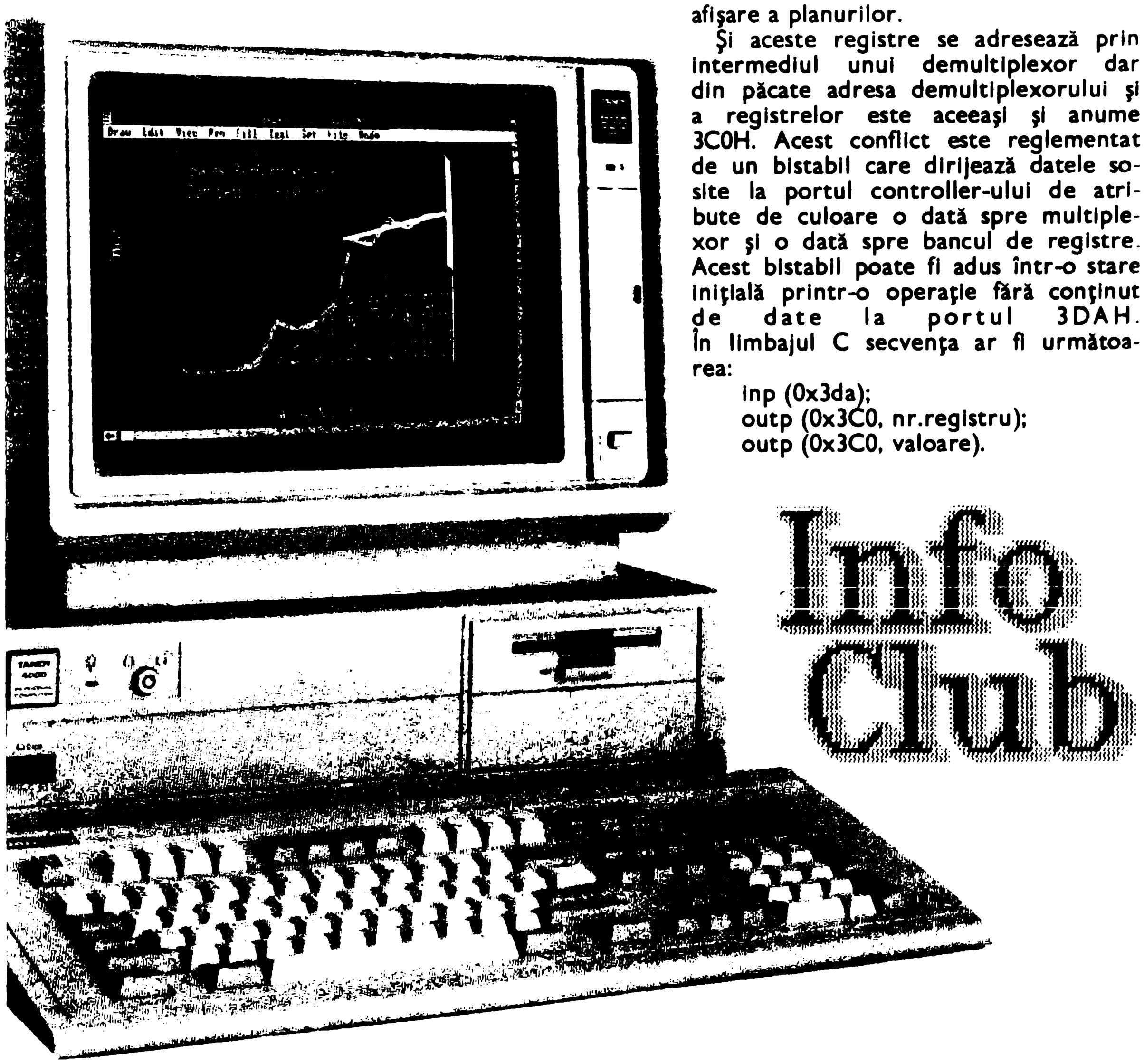
Astfel, prin schimbarea numărului de octeţi citiţi pe o linie, a numărului de linii şi a timpului de blancare se pot obţine două moduri de lucru nestandard. Primul constă în forţarea rezoluţiei pînă la 800x600 dar numai unele monitoare mai sînt capabile să sincronizeze. Un monitor care nu se poate sincroniza este supus unor regimuri tranzitorii periculoase în blocul de baleiaj aşa că nu recomandăm astfel de experimente. Alt mod de lucru nestandard constă în afişarea unei ferestre 640x350 care poate fi deplasată pe un desen realizat pe un spaţiu 800x600. Acest ultim mod este exemplificat în binecunoscutul program EGADEMO la imaginea tomografică. Mult mai simplu de realizat este o fereastră 640x350 (640x480-VGA) care glisează pe un desen 640x720. Acest lucru se obţine prin modificarea valorii registrelor care specifică linia de unde începe afişarea (registrele 0CH pentru high/0DH pentru low).

Graphics Controller. Are în compunere 9 registre care răspund la cele mai importante funcţii de grafică:

- masca de culoare pentru scrierea pixelilor, adică culoarea de trasare;
- masca de protecţie a unor planuri;
- selecţia unei culori de comparaţie;
- selecţia funcţiilor ALU şi rotaţia octetului de date;
- selecţia unui plan pentru citire;
- selecţia modurilor de citire şi

scriere;

- funcţii de lucru pe 16 biţi simultan în două planuri;
- mască de planuri invalidate la comparaţie;
- mască de scriere a octetului.



Attribute Controller.

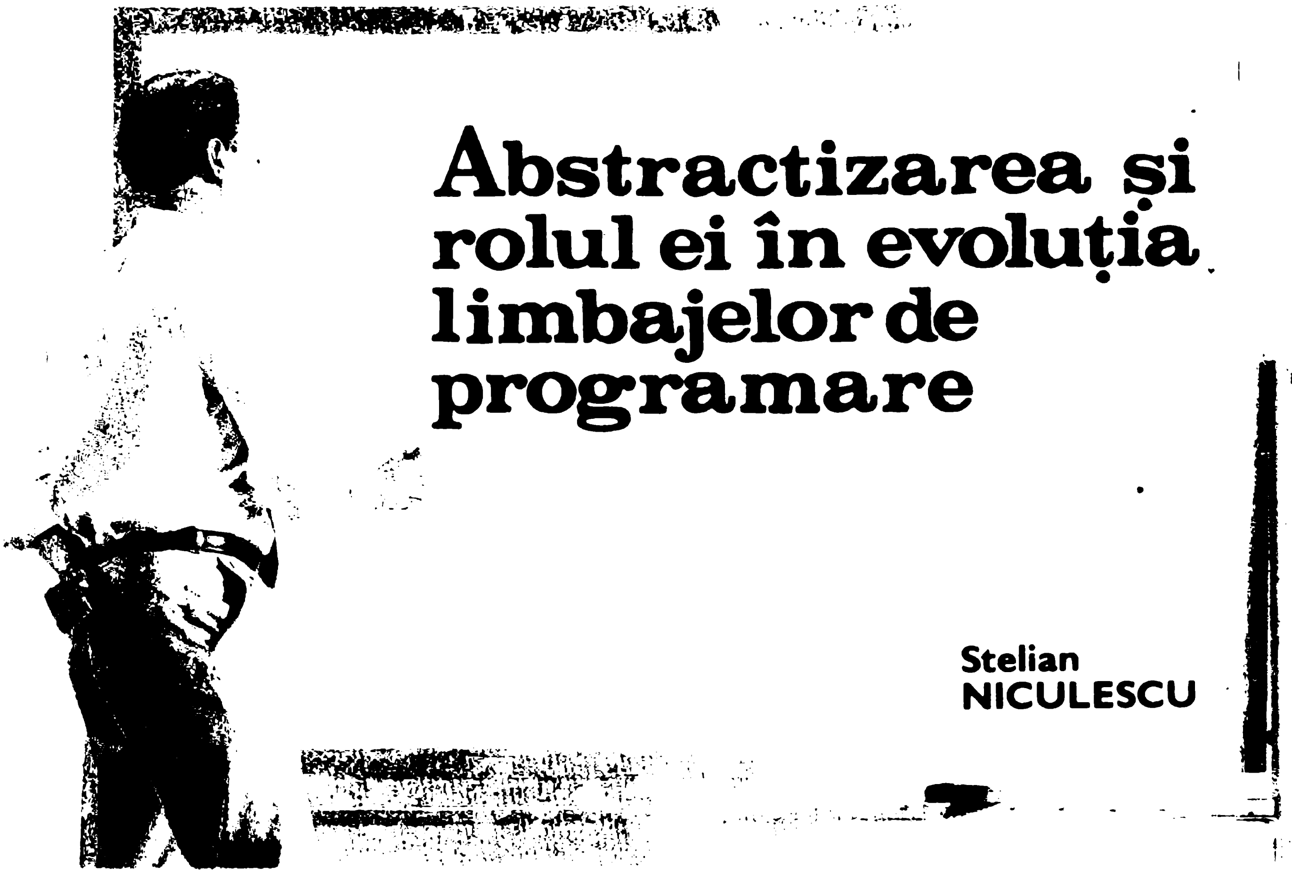
Conţine 19 registre cu rol direct în controlul culorilor. Primele 16 registre sînt de fapt Tabela de Culori, iar următoarele se ocupă cu: controlul modului de lucru (grafic/alfanumeric); controlul culorii chenarului; masca de afişare a planurilor.

Şi aceste registre se adresează prin intermediul unui demultiplexor dar din păcate adresa demultiplexorului şi a registrelor este aceeaşi şi anume 3C0H. Acest conflict este reglementat de un bistabil care dirijează datele sosite la portul controller-ului de atribute de culoare o dată spre multiplexor şi o dată spre bancul de registre. Acest bistabil poate fi adus într-o stare iniţială printr-o operaţie fără conţinut de date la portul 3DAH. În limbajul C secvenţa ar fi următoarea:

```
inp (0x3da);
outp (0x3C0, nr.registru);
outp (0x3C0, valoare).
```

Cele prezentate nu au avut rolul să dea o informaţie completă, organizată în genul unui breviar, ci să vă familiarizeze cu nişte principii generale care, în acest caz, sînt mai greu de desprins din literatură. Documentaţia de firmă este extrem de condensată şi fără o pregătire prealabilă e greu de parcurs. O alternativă ar fi cartea scrisă de Richard F. Ferraro — Programmer's Guide to the EGA — VGA Oard (Addison-Wesley/July 1988), cu toate că este exagerat de detaliată şi cele exact 600 de pagini fac ca esenţialul să fie greu de sintetizat la prima parcurgere.

Dacă am reuşit să stîrnim interesul dumneavoastră, într-un episod următor putem prezenta o bibliotecă grafică pentru limbajul C cu exploatarea tuturor facilităţilor cuplorului EGA (VGA)!



Abstractizarea şi rolul ei în evoluţia limbajelor de programare

Stelian NICULESCU

Familia limbajelor de programare (FLP) se situează, ca evoluţie, între limbajul binar (LB, propriu calculatoarelor electronice) şi limbajul natural (LN). În timp ce LB reprezintă limita stîngă a FLP, avînd în vedere evoluţia în timp, fiind primul membru al FLP, LN este limita dreaptă, continuînd să constituie încă un ideal greu de atins. Nu putem să nu observăm că un salt calitativ remarcabil în evoluţia FLP este datorat apariţiei generaţiei a V-a de calculatoare, limbajele de generaţiile I - IV fiind de tip imperativ şi cu elemente descriptive, iar cele de generaţia V fiind de tip imperativ şi cu elemente imperative. Iată saltul de la **CUM** la **CE** în activitatea de programare. Ar mai fi de menţionat faptul că limbajele noi care apar se constituie în restricţii ale limbajului natural, din ce în ce mai apropiate de acesta.

În cele ce urmează vom face unele consideraţii privind rolul abstractizării în evoluţia în timp a FLP, rol pe care îl considerăm deosebit de însemnat.

O abstractizare a unui obiect reprezintă o caracterizare a acestuia prin nişte atribute, ceea ce duce la o clasă de echivalenţă, în care elementul respectiv este membru. Obiectele clasei sînt numite realizări, implementări, rafinamente sau instanţieri ale abstractizării. În fapt, o abstractizare este concepută ca o specificare de tip **CE**, în timp ce implementările ei sînt asociate cu specificaţii de tip **CUM**, procesul de abstractizare prezentînd interes practic numai în condiţiile în care specificarea de tip **CE**, caracterizînd atributele esenţiale ale unui obiect, este substanţial mai simplă decît specificaţia de tip **CUM**.

Abstractizarea are un rol deosebit în studiul şi asigurarea modularităţii programelor, fiecare modul avînd o specificare de tip **CE** (abstractizare), prin care se precizează ce se face, precum şi una de tip **CUM** (rafinament, realizare sau implementare), care materializează modul de realizare a specificaţiei **CE**.

Alături de abstractizare există două principii care duc la sporirea rolului specificaţiilor de tip **CE**, în detrimentul celor de tip **CUM**, ceea ce concordă cu noile viziuni în ingineria programării: principiul localizării informaţiilor implicate în realizarea abstractizării; principiul lui Parnas, cunoscut sub denumirea „ascunderea informaţiei", avînd drept scop să facă „vizibile" numai acele proprietăţi ale modulelor care sînt strict necesare în interfaţarea lor.

În procesul de programare sînt cu-

D.P. TIBERIU

PLACA DE BAZĂ IBM PC-XT

Placa de bază **(mother board)** a calculatorului PC-XT poate fi considerată ca un calculator de 16 biţi de utilizare generală. După conectarea echipamentelor periferice şi a comenzilor acestora rezultă un **calculator personal profesional.** Firma IBM a proiectat familia de calculatoare PC ca nişte sisteme deschise, care se pot dezvolta prin conectarea la magistrala universală a modulelor suplimentare. Deoarece IBM a publicat datele tehnice ale magistralei, alte firme au putut produce atît plăcile de bază cît şi modulele funcţionale **(add-on cards).**

Placa prezentată de noi în acest poster este denumită **MEGABOARD.** Aceasta conţiine circa 100 circuite integrate, opt conectoare pentru add-on cards şi se poate împărţii (diviza) în următoarele blocuri funcţionale:

- blocul procesorului (8088, 8087) şi magistrala locală;
- bloçul accesului direct la memorie (circuit DMA-8237);
- magistrala de extensie (Expansion Bus);
- blocul de comandă al magistralei externe (External Bus Control);
- blocul de memorie cu acces aleator (Random Acces Memory);
- blocul memoriei de tip ROM (Read Only Memory);
- blocul echipamentelor de intrare-ieşire (Peripherals).

Cele trei magistrale sînt: de sistem, externă şi de extensie. Acestea pot fi comandate de procesor sau de comanda DMA.

În cele ce urmează ne vom opri în detaliu asupra unor blocuri funcţionale, mai importante.

Blocul procesorului. Elementele de bază ale blocului sînt procesorul 8088 varianta de 8 biţi, sau 8086 pentru 16 biţi — şi coprocesorul numeric 8087 (bloc 1). Circuitul 8288 (U19 bloc 4) decodifică semnalele de stare ale procesorului, S0/, S1/, S2/ şi generează semnalele de comandă: IOW/, IOR/, MEMW/, MEMR/, MWTC/, INTA/ şi, suplimentar, semnale de comandă ale buffer-ului magistralei de date U1: ALE, DEN, DT-R.

Semnalele de tact propriu din circuitul 8284 (U3 — bloc 2) comandat de un cuarţ de 14,31818 MHz. Condensatorul C8 permite fixarea riguroasă a frecvenţei. Circuitul 8284 are trei ieşiri: OSC (frecvenţa f) — accesibil doar pe magistrala de extensie; CLK 88 (frecvenţă f/3) — semnalul de tact de bază pentru procesor şi majoritatea circuitelor de pe placa de bază; PCLK (frecvenţă f/6) — folosit pentru citirea tastaturii şi, după împărţirea la doi în circuitul U35 — ca semnal de tact pentru contorul 8253.

După conectarea alimentării sau acţionarea comenzii S1-RESET, procesorul şi alte circuite sînt şterse cu semnalele RES 88 şi RESET/Z din circuitul U3. Blocul format din R1-R2-C12 şi CR1 asigură o durată suficient de mare a semnalului de ştergere. Pentru conversia liniilor locale ADO-AD7 şi AA12-A-A19 sînt folosite buffer-ele LS373 (U2 şi U7 — bloc 5), iar liniile AA8 şi AA11 folosesc jumătate din U4-LS244. Liniile locale de date sînt legate de U1-LS245. Aceste circuite trec în stare de impedanţă ridicată în momentul preluării comenzii de către DMA (8237).

Circuitele 8088, 8087, 8237 pot trece în stare de aşteptare (wait state) dacă un echipament oarecare input-output sau memoria necesită un timp îndelungat de acces. Circuitul special de generare a semnalului de confirmare READY construit din bistabilii U36, U37, U44 şi porţi (bloc 3), comandînd intrarea AEN1/ a circuitului U3-8284, impune starea de aşteptare a procesorului.

Încetinirea lucrului procesorului depine de tipul transmisiei, care poate fi:

1.scriere sau citire date de către procesor în şi din memorie;
2. scriere sau citire date de către procesor în şi din echipamentele in-out (periferice);
3. acces direct la memorie (DMA).

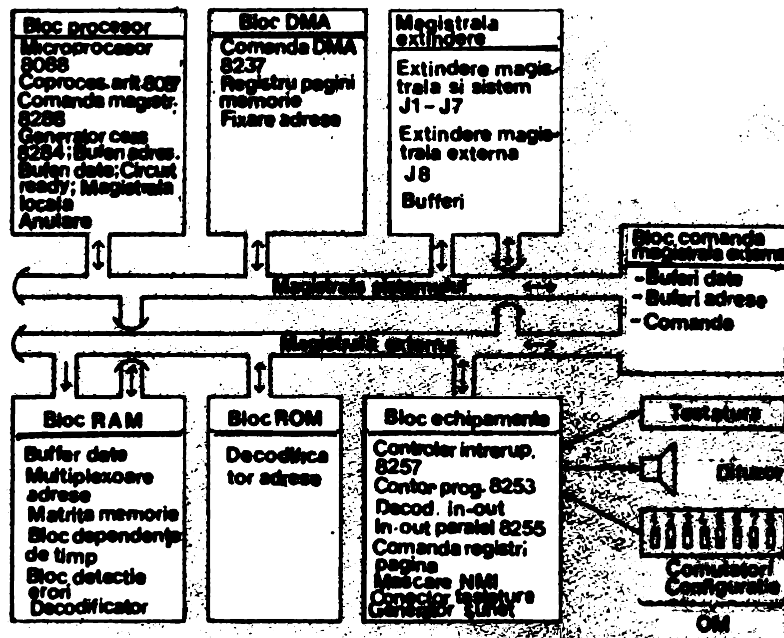
În primul caz, dacă memoria se află pe placa de bază, nu trebuie introdusă starea de aşteptare. Accesul la memoria rezidentă pe modulele suplimentare poate fi încetinit prin impunerea nivelului jos de tensiune pe linia I-OCHRDY a magistralei de extensie. În al doilea caz, transmisia este întotdeauna prelungită cu un ciclu de aşteptare. Dacă echipamentul in-out se găseşte pe modulul extern (exemplu pe cel de comandă al floppy) transmisia poate fi suplimentar întîrziaă prin linia I-OCHRDY.

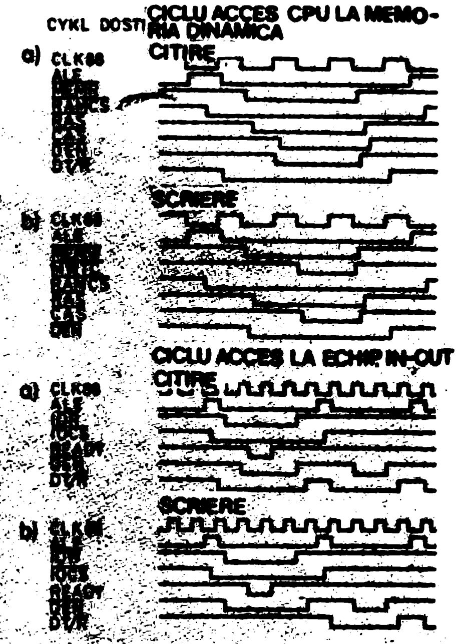
Procesorul trimite sau recepţionează date, realizînd operaţiile: scriere-citire în sau din memorie; scriere-citire în sau din echipamentele in-out; confirmarea acceptării întreruperii.

Operaţiunea de citire a memoriei de către procesor (detaliată în figură) se poate împărţi în următoarele etape:

- trimiterea către comanda magistralei 8288 a informaţiei privind începerea operaţiei de citire (liniile S0, S1);
- plasarea pe liniile locale a adresei celulei respective de memorie;
- decodificarea în comanda 8288 a stării, transmiterea semnalului ALE şi fixarea adresei pe magistrala de sistem (circuitele U2, U4, U7 — bloc 5), bufferarea adresei pe magistrala externă (U8, U9, U10 — bloc 9);
- deblocarea bufferului de date U1 şi fixarea direcţiei de transmitere de pe magistrala de sistem pe magistrala locală;
- generarea semnalului MEMR/ şi transformarea lui U22(bloc 10) în semnal de citire XMEMR/;
- decodificarea adresei de către toate circuitele de memorie;
- plasarea datei pe magistrala corespunzătoare (în cazul memoriei interne ROM conectată la magis-

Spot



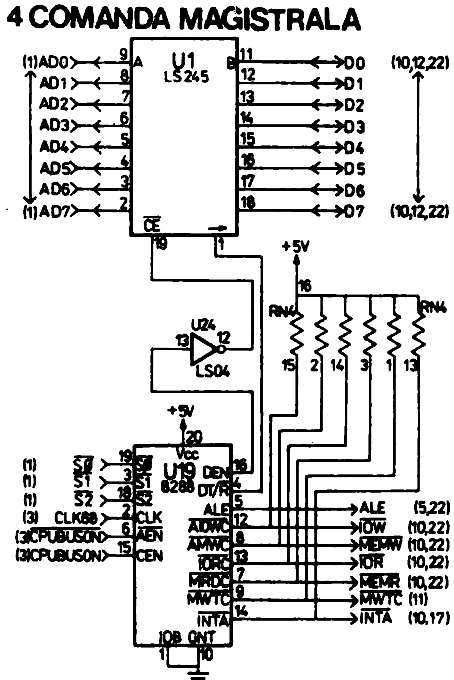






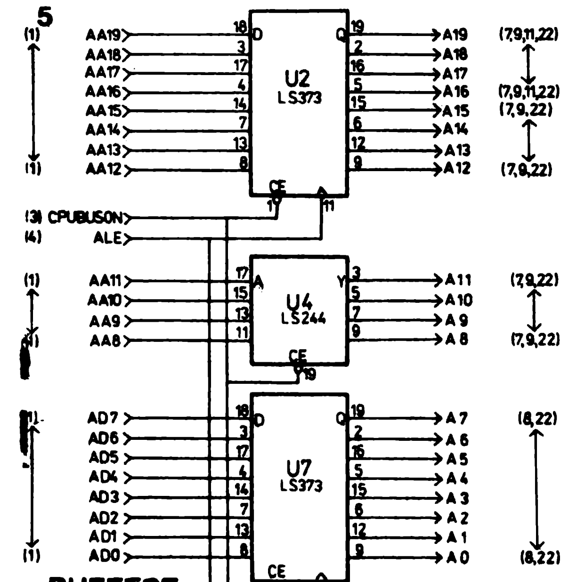


4 COMANDA MAGISTRALA



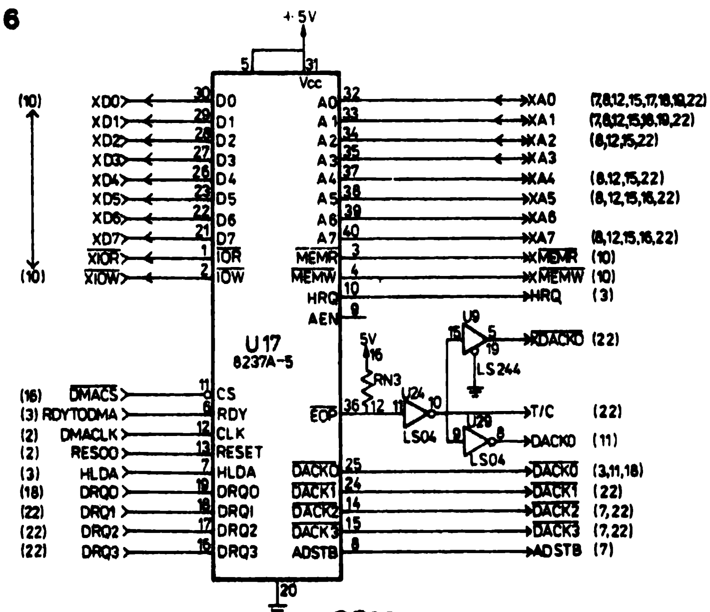
Spot

5



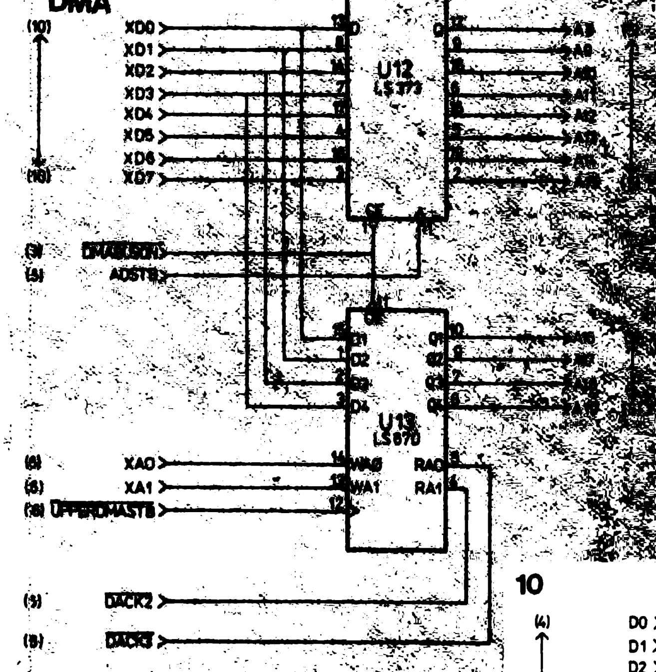
BUFFERE SI ADRESE SISTEM

6

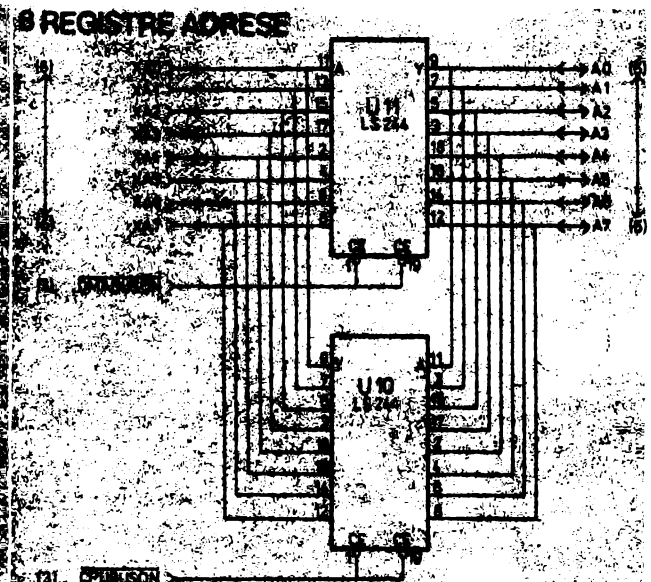


COMANDA DMA

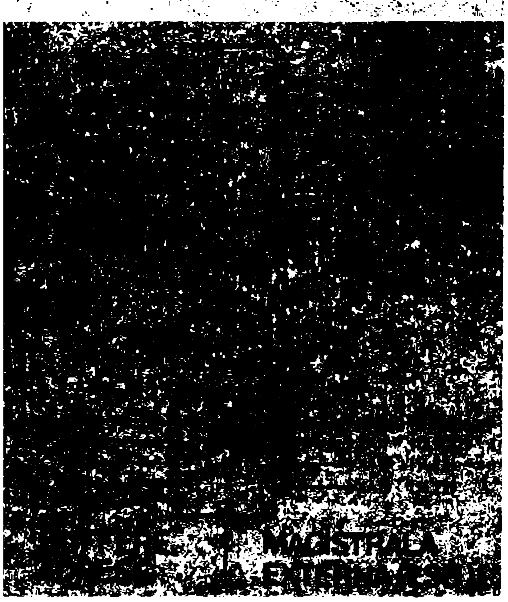
7 REGISTRE ADRESE DMA



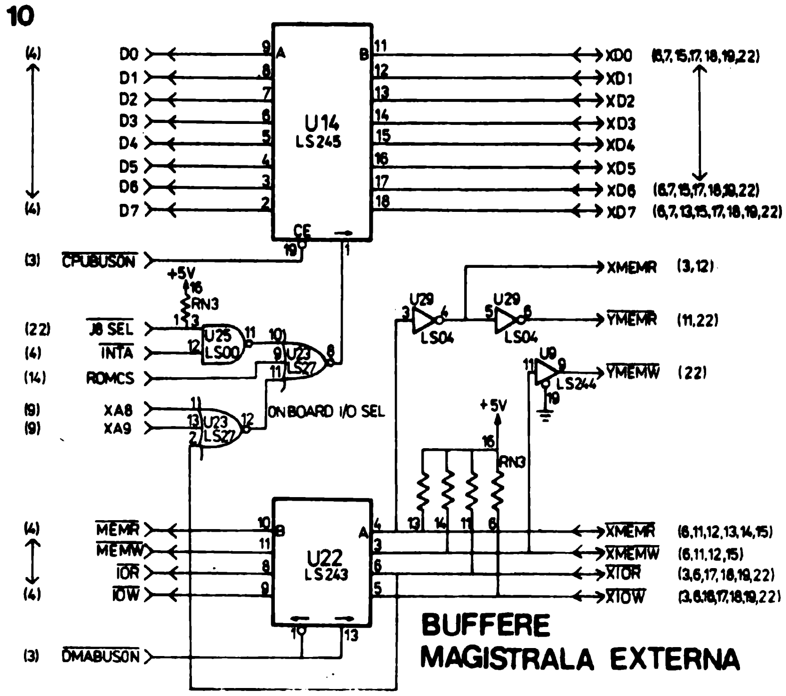
8 REGISTRE ADRESE



MAGISTRALA EXTERNA



10



BUFFERE MAGISTRALA EXTERNA

trala externă acţionează buffer-ul Ul4 — bloc 10);

● eventuala prelungire a ciclului prin iniţierea ciclului de aşteptare (operaţia de citire constă din minimum 4 cicluri de ceas ale procesorului — respectiv 840 ns).

Blocul DMA

Blocul DMA realizează două funcţii: împrospătarea ciclică a memoriei (refresh memory) şi transmiterea rapidă între echipamentele in-out conectate la magistrala de extensie şi memoria sistemului.

Elementul de bază al modului este circuitul 8237 A-5 (U7 — bloc 6). Acest circuit foloseşte două registre auxiliare U12-74LS373 şi U13-74LS670 (bloc 7).

Registrul U12 trimite cei 8 biţi de adresă A8-A15 indicaţi de comandă pe magistrala externă de date. Împreună cu biţii A0-A7 generaţi din nou la transmiterea fiecărui octet, se formează cei 16 biţi ai adresei. Registrul este încărcat la începutul fiecărei transmisii şi reactualizat după generarea transferului de către contorul care creează semnalele A0-A7. Circuitul U13 extinde adresa DMA cu 4 biţi necesari pe magistrala de 20 de biţi a microprocesorului 8088. Acest circuit este plasat în zona de adresare in-out şi ocupă în aceasta trei adrese (81H, 82H şi 83H) corespunzătoare celor trei canale DMA (2, 3 şi 1). În aceste adrese se pot scrie valorile de 4 biţi care apoi, în timpul operaţiilor DMA sînt transmise pe liniile de adrese A16-A19. Alegerea celor 4 biţi în adresa respectivă este comandată de liniile DACK2/ şi DACK3/. La transmitere în canalul 2 este înscrisă valoarea memorată în 81H, în canalul 3-82H şi în 1-83H. La transmiterea în canalul 0, circuitul U13 nu este folosit.

În timpul transmisiei DMA se efectuează trimiterea de date între echipamentele in-out şi memorie, conform următoarei scheme:

1. Programul de execuţie iniţiază canalul DMA conectat cu echipamentul respectiv in-out, scriind în comanda DMA 16 biţi mai puţin semnificativi ai adresei de început a bufferului de memorie iar în registrul U13 cei 4 biţi semnificativi. De asemenea, în comanda DMA este scrisă cifra octeţilor trimişi şi instrucţiunea de citire sau scriere;

2. Comanda aşteaptă pînă cînd echipamentul transmite pe magistrala de extensie semnalul DRQn, semnalizînd pregătirea pentru transmisie;

3. Comanda răspunde cu semnalul HRQ care pentru procesor constituie cererea de a permite comanda magistralei;

4. Dacă magistrala nu este blocată (semnalul LOCK/ nu este activ) procesorul trimite semnalul HLDA şi permite astfel comenzii DMA preluarea comenzii magistralei. Semnalul HLDA este generat de U30, U44 şi U36;

5. Comanda scrie biţii de adresă A8-A15 în registrul U12 şi fixează liniile A0-A7; trecînd „0" pe liniile DACKn/ informează echipamentele externe că va avea loc o transmisie;

6. Bistabilul U44 (care generează semnalul CPU BUSON) decuplează de pe magistrala de sistem generatoarele de linii de adresare, linii de date şi linii de comandă , cu influenţă asupra circuitelor U1, U19, U2, U4 şi U10. Prin U9-7 şi semnalul AEN la magistrala de extensie, bistabilul informează echipamentele in-out că pe durata transmisiei DMA nu mai trebuie să decodifice adresele magistralei. Semnalul CPU BUSON blochează şi decoderele plăcii de bază şi toate ehipamentele in-out rezidente pe aceasta;

7. Cu semnalele DMA BUSON (circuitul U44) sînt activate registrele de adrese şi semnalele de comandă (circuitele U12, U13 — bloc 7, U11, U22 — bloc 8). Semnalul DMA BUSY/ (U44-7) opreşte procesorul de a prelua comanda magistralei;

8. Următorul pas depinde de felul de transmisie. La citirea datelor din memorie comanda DMA generează semnalul XMEMR/, iar memoria adresată trimite data pe magistrala de sistem. In următorul ciclu comanda DMA generează semnalul XIOW/ scriind data în echipamentul activat cu senmalul DACKn. La scrierea datelor în memorie comanda generează semnalul XIOR/, iar echipamentul activat de DACKn/ trimite data pe magistrala de sistem. În următorul ciclu comanda generează semnalul XMEMW/ scriind data în celula adresată.

În orice caz transmisia este prelungită cu minimum un ciclu de aşteptare. Semnalele XIOR/, XIOW/ prin U31, U37, U44 formează semnalul de „gata" al canalului 1, 2 sau 3 — RDYTODMA.

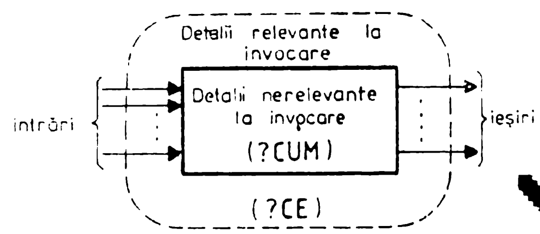
9. Semnalele de comandă HRQ şi DACKn/ sînt retrase;

10. Circuitul U44 anulează semnalul CPU BUSON/ şi blochează generatoarele canalului DMA şi activează bufferii modulului procesor;

11. Este anulat semnalul DMABUSY/ şi dacă procesorul se găsea în stare de aşteptare la terminarea transmisiei DMA, trece acum în funcţionare normală.

noscute trei tipuri de abstractizări, corespunzătoare celor trei componente conceptuale fundamentale ale unui limbaj: abstractizarea procedurală, abstractizarea datelor, abstractizarea controlului.

Abstractizarea procedurală sau funcţională a fost cel mai bine suportată de limbajele convenţionale, ea corespunzînd conceptului clasic de subrutină (procedură). La momentul invocării subrutina se poate trata ca o „cutie neagră", fiind separate cele două tipuri de detalii, aşa cum se ilustrează în figura alăturată.



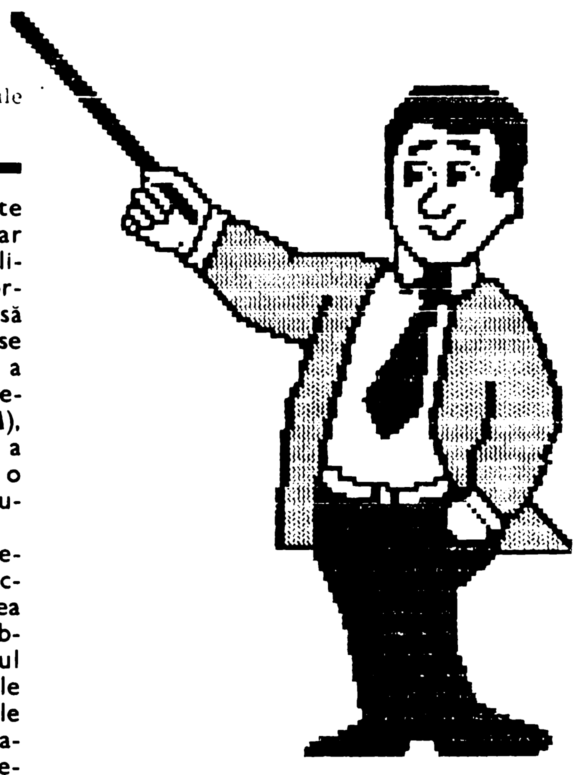
Principiul abstractizării procedurale (funcţionale)

Abstractizarea datelor permite programatorului să se preocupe doar de comportamentul obiectelor implicate în program, adică de tipul informaţiei care poate fi stocată sau extrasă (detalii de tip **CE**). El nu trebuie să se preocupe de modul de reprezentare a obiectelor şi nici de algoritmii de memorare sau acces (detalii de tip **CUM**), abstractizarea datelor avînd rolul de a se amîna astfel de decizii pînă la o etapă ulterioară proiectării programului.

O **abstractizare a controlului** presupune o metodă de secvenţiere a acţiunilor dintr-un program, majoritatea limbajelor de nivel înalt asigurînd abstractizări predefinite de tipul **if/then/else** şi **while/do.** Limbajele mai noi permit în plus abstractizări ale controlului concepute de programator, acestea reprezentînd de fapt generalizări ale metodelor de secvenţiere repetitivă disponibile în majoritatea limbajelor de programare.

Abstractizările se regăsesc în limbajele de programare fie sub formă implicită, fie sub formă explicită. Cele implicite sînt irevocabile (structuri de date predefinite, structuri de control), iar cele explicite (sau extinse) sînt introduse (concepute) de programator, limbajul trebuind să furnizeze mecanisme speciale pentru definirea de noi abstractizări, ceea ce conferă limbajului facilităţi de extensibilitate.

Se apreciază că majoritatea limbajelor convenţionale conţin prea multe abstractizări implicite şi prea puţine (sau insuficient de generale) mecanisme pentru definirea explicită de abstractizări. Abstractizările implicite constituie pentru multe dintre limbajele de programare o barieră artificială pentru programator, atît din punct de vedere al utilizării, cît şi din punct de vedere conceptual. Introducerea de abstractizări explicite presupuse ca limbajul să conţină mecanisme lingvistice corespunzătoare, atît pentru definirea şi extinderea lor, cît şi pentru utilizare, aceasta fiind o cale de urmat.



Încheiem scurta noastră intervenţie prin a observa că viitorul este al lui **CE** în detrimentul lui **CUM,** ceea ce duce practic la prosperitatea limbajelor PASCAL, C, LISP, PROLOG ş.a.

Valeriu
IORGA

Utilizarea sistemului TURBO Pascal

TURBO Pascal este un produs deosebit de reuşit al firmei Borland, realizat în anul 1983 de către Philippe Kahn. TURBO Pascal nu este un simplu compilator, ci un mediu de programare interactiv, extrem de productiv, asigurînd suport pentru compilarea, execuţia, depanarea şi testarea programelor Pascal.

Versiunea 3.0 este implementată atît pe microcalculatoare de 8 biţi realizate cu microprocesoare Z80 sub sistemul de operare CP/M-80, cît şi pe microprocesoare de 16 biţi sub o diversitate de sisteme de operare: MS-DOS, PC-DOS, CP/M-86. Versiunile ulterioare: 4.0, 5.0, 5.5 au fost dezvoltate numai pe microcalculatoare cu 16 biţi.

TURBO Pascal urmăreşte fidel limbajul Pascal „standard" înţelegînd prin aceasta nu atît standardul ANSI Pascal/IEEE770.X3 97-1983 cît „Pascal User Manual and Report" a lui K. Jensen şi N. Wirth.

Prezentarea pe care v-o propunem este o prelucrare a volumului „TURBO Pascal — version 3.0 — Reference Manual" şi are în vedere numai particularităţile specifice ale limbajului şi operarea în TURBO Pascal, presupunînd cunoscut limbajul Pascal standard (o prezentare a acestuia este făcută în revista „Ştiinţă şi Tehnică", începînd din nr. 7/1989).

Mediul de programare TURBO Pascal

Fişiere necesare utilizării sistemului Turbo Pascal.

TURBO.COM — conţine compilatorul şi editorul TURBO Pascal;
TURBO.OVR — necesar numai sub CP/M-80 pentru execuţia fişierelor .COM din TURBO;
TURBO. MSG — conţine un fişier text cu mesajele de eroare ale compilatorului;

Pentru utilizarea sistemului TURBO Pascal este suficient numai fişierul TURBO.COM care ocupă 16 kocteţi. Celelalte două fişiere ocupă 33 kocteţi, respectiv 1.5 kocteţi.

Fişiere necesare instalării sistemului

TINST.COM — conţine programul care adaptează sistemul TURBO Pascal (programul TURBO.COM) pe un anumit tip de calculator şi de terminal prin introducerea unor parametri de instalare;
TINST.DTA — conţine parametrii de instalare pentru diferite tipuri de terminale;
TINST.MSG — conţine un fişier text cu mesajele programului de instalare.

Lansarea sistemului TURBO Pascal

După introducerea discului conţinînd fişierele TURBO Pascal în unitatea curentă se face lansarea sistemului în execuţie prin comanda: **TURBO<CR>**. Pe ecranul calculatorului se afişează un mesaj specificînd versiunea, sistemul de operare şi se interoghează utilizatorul (prin Y/N) dacă să încarce sau nu mesajele de eroare (fişierul TURBO.MSG).

```
TURBO-Pascal System          Version N.NNX
                                  [System]
Copyrigh (C 1983, 1984, by BORLAND inc.
No terminal selected
Include error message (Y/N)? [ ]
```

În continuare pe ecran apare meniul principal.

Meniul principal

Precizarea comenzilor disponibile:

```
Logged drive: A

Work file:
Main file:

Edit   Compile   Run        Save
Dir    Quit      compiler   Options

text:  O bytes
free:  62903 bytes
```

Comenzile se introduc prin iniţiala lor (exceptînd comanda execute). Comanda selectată este executată imediat, meniul dispărînd de pe ecran; pentru a-l face să reapară se introduce <CR>.

Pentru schimbarea unităţii de discuri curente se dă comanda L şi la răspunsul sistemului **New drive:** se introduce noua unitate selectată (de exemplu B:<CR>).

Fişierul de lucru (folosit la editare, compilare, execuţie) este specificat prin comanda W şi la răspunsul sistemului **Work file name:** se introduce numele fişierului (de exemplu TEST.PAS). Numele fişierului poate avea sau nu extensie, în acest din urmă caz fişierul primind în mod automat extensia .PAS. Nu se recomandă folosirea extensiilor .COM, .CHN, .CMD, .BAK întrucît ele au destinaţii speciale. Fişierul specificat, dacă există pe disc, este citit în memorie, iar în caz contrar apare mesajul **New File.**

Comanda M permite numirea unui fişier principal care să conţină directive de includere (•$I nume•) de fişiere; acesta este încărcat în memorie în locul fişierului de lucru, care este salvat. La detectarea unei erori la compilare, în fişierul principal sau în fişierul inclus, fişierul ce conţine eroarea este încărcat în memorie şi devine fişier de lucru. Se permite astfel corectarea erorii prin editare, după care se face comutarea fişierelor în memorie şi se reia compilarea.

Comanda E lansează operaţia de editare asupra fişierului de lucru. Meniul principal dispare şi este activat editorul. Editorul TURBO este o variantă simplificată a editorului WordStar.

Comanda C lansează compilarea fişierului principal. Dacă nu este specificat un fişier principal este compilat fişierul de lucru. Compilarea poate fi întreruptă în orice moment prin apăsarea oricărei taste. Din compilare poate rezulta un program rezident în memorie, sau un program rezident pe disc într-un fişier .COM sau într-un fişier .CHN, în funcţie de opţiunile de compilare specificate.

Comanda R lansează în execuţie un program rezident în memorie sau un fişier .COM sau un fişier .CMD.

Comanda X (execute) prezentă în meniul principal numai în sistemul de operare CP/M-80 permite lansarea în execuţie a unui program (fişier .COM) din meniul principal TURBO.

Comanda S salvează pe disc fişierul de lucru. După compilare se recomandă salvarea fişierului sursă rezultat din editare.

Comanda D listează catalogul unităţii curente şi precizează spaţiul neutilizat de pe discul curent. La introducerea comenzii D se afişează mesajul **Dir mask: [ ];** pentru listarea catalogului se tastează <CR>; listarea unui singur nume TURBO sub sistemul de operare, după ce în prealabil utilizatorul e întrebat dacă doreşte salvarea fişierului de lucru. de fişier sau a unui grup de fişiere se face indicînd un nume de fişier sau o mască conţinînd caracterele • şi ?.

Comanda Q face ieşirea din sistemul

Comanda O selectează un meniu în care se pot modifica opţiunile implicite ale compilatorului şi asigură o funcţie de localizare a erorilor la execuţie din programele compilate. La introducerea acestei comenzi apare un nou meniu privind modul de lucru al compilatorului avînd forma:

compile — **M**emory
Com-file
c**H**n-file
command line **P**arameter:
Find run-time error **Q**uit

Comenzile M, C, H selectează destinaţia unde se depune codul rezultat din compilare (implicit în memorie). În cazul în care codul este rezident în memorie, el poate fi lansat în execuţie prin comanda Run.

Comanda C depune codul generat (şi biblioteca Pascal) într-un fişier .COM pe disc. Lansarea în execuţie se face în acest caz prin numele fişierului.

Comanda H depune codul generat (fără biblioteca Pascal) într-un fişier .CHN pe disc. Lansarea în execuţie se face din alt program TURBO Pascal folosind procedura Chain. Dacă în timpul execuţiei unui program compilat în memorie se produce o eroare la execuţie, este apelat în mod automat editorul şi eroarea este afişată. La execuţia unui fişier .COM sau .CHN apariţia unei erori la execuţie duce la afişarea codului erorii şi a contorului programului. Localizarea erorii în textul sursa se face folosind *comanda F* specificînd ca parametru adresa de oprire.

Editorul de texte TURBO Pascal

Este un editor orientat pe ecran foarte asemănător editorului WordStar prezentat sub forma unui breviar de comenzi şi în paginnile revistei „Ştiinţă şi tehnică" (1988). El poate fi lansat din meniul principal folosind comanda E.

Diferenţe dintre editorul TURBO faţă de WordStar

1) Comenzile de deplasare CTRL S şi CTRL D nu permit trecerea de la o linie la alta.

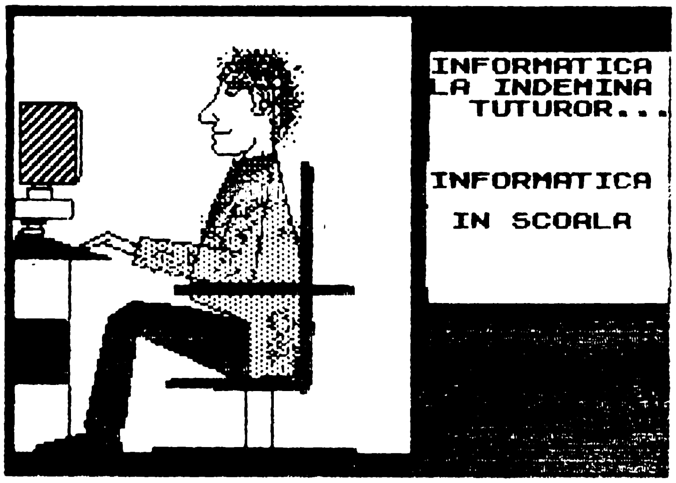
2) Comanda CTRL KT de marcare a unui singur cuvînt ca un bloc este o comandă specifică editorului TURBO.

3) Comanda CTRL KD termină editarea revenind în mediul principal (în WordStar această comandă face şi salvarea fişierului pe disc).

4) Comanda CTRL QL reface o linie cît timp cursorul nu a părăsit-o.

5) Comanda TAB pune tab-stopuri în linia curentă pe începuturile de cuvinte din linia precedentă.

6) Comanda CTRL QI de activare/inhibare autoindentare este specifică editorului TURBO.



Directive ale compilatorului TURBO Pascal

Acestea se introduc sub forma unor comentarii cu sintaxă specială, ca de exemplu:
(•$B—,R+,V—•)
Toate directivele au valori implicite, alese astfel încît să optimizeze viteza de execuţie şi să minimizeze codul generat. Valorile implicite ale opţiunilor sînt cele subliniate.

a) Directive comune tuturor sistemelor de operare

1) *B*+ dispozitivul consolă (CON:) este asignat fişierelor standard Input şi Output
   B— dispozitivul terminal (TRM:) este asignat fişierelor standard Input şi Output
2) *C*+ citire CTRL C întrerupe execuţia programului
   CTRL S activează/inhibă afişarea pe ecran
   C— CTRL C nu este interpretat
3) *I*+ tratează erorile în operaţiile de intrare/ieşire
   I— tratarea erorilor este făcută de programator
4) I nume-fişier — include în programul compilat fişierul cu numele specificat
5) R+ verificarea la execuţie a încadrării indicilor tablourilor în dimensiuni
   R— nu se face nici o verificare asupra indicilor
6) *V*+ verificarea congruenţei listelor de parametri actuali şi formali
   V— nu se face nici o verificare
7) U+ permite suspendarea execuţiei programului prin CTRL C
   U— execuţia programului nu poate fi oprită

b) Directive ale compilatorului TURBO Pascal sub sistemele PC-DOS şi MS-DOS.

1) Gn defineşte bufferul fişierului standard de intrare Input
   GO — fişierul Input se referă la dispozitivul CON: sau TRM:
   Se plasează înaintea părţii declarative.
2) Pn defineşte bufferul fişierului standard de ieşire Output
   PO — fişierul Output se referă la dispozitivul CON: sau TRM:
3) *D*+ interoghează asupra stării unui fişier text deschis prin Reset, Rewrite sau Append. Dacă fişierul este un dispozitiv, atunci operaţiile de intrare/ieşire se vor face caracter cu caracter (fără bufferare)
   D— nu se face nici o verificare şi operaţiile se fac bufferat
4) Fn controlează numărul de fişiere care pot fi deschise simultan (implicit F16).
   Se plasează înaintea părţii declarative.

c) Directive ale compilatorului comune sub sistemele PC-DOS, MS-DOS, CP/M-86

1) *K*+ generează cod de verificare pentru alocarea stivei (se verifică dacă este spaţiu suficient în stivă pentru variabilele locale, înaintea fiecărui apel de subprogram).

d) Directive ale compilatorului sub sistemul CP/M-80

1) *A*+ controlează generarea de cod absolut nerecursiv (nu sînt permise apeluri recursive)
   A— sînt permise apeluri recursive
2) Wn controlează numărul nivelurilor de îmbricare a instrucţiunilor with (cîte înregistrări pot fi deschise în interiorul unui bloc $1 \leq n \leq 9$; implicit W2)
3) *X*+ controlează generarea de cod care să asigure accesul la tablouri cu viteza optimă

Diferenţe între TURBO Pascal şi standard

Extensii TURBO faţă de standard

1). Simboluri suplimentare:
— litere: _ (subliniere)
— simboluri speciale : **$**
— simboluri cuvinte :

● Ghidul utilizatorului

absolute	**external**	**inline**	**overlay**
shl	**shr**	**string**	**xor**

— identificatori standard:

Addr	Assign	Aux	AuxInPtr	AuxOutPtr
BlockRead	BlockWrite	BufLen	Byte	Chain
Close	ClrEOL	ClrScr	Con	ConInPtr
ConOutPtr	Concat	ConstPtr	Copy	CrtExit
CrtInit	DelLine	Delay	Delete	Erase
Execute	Exit	FilePos	FileSize	FillChar
Flush	Frac	GetMem	GotoXY	Halt
HeapPtr	Hi	IOresult	InsLine	Insert
Int	Kbd	KeyPressed	Length	Lo
LowVideo	Lst	LstOutPtr	Mark	Mem
MemAvail	Move	NormVideo	Pi	Port
Pos	Ptr	Random	Randomize	Release
Rename	Seek	SizeOf	SeekEof	SekEoln
Str	Swap	Trm	UpCase	Usr
UsrInPtr	UsrOutPtr	Val		

2) tipuri standard suplimentare:
byte=0...255;
string[n]=**array**[0...n] **of** char;

3) Caractere de control:

ValASCII {caracterul avînd valoarea ASCII specificată}
^Caracter {Control-Caracter}

Secvenţele de caractere de control se concatenează fără separatori între ele.

4) Secţiunile din partea declarativă pot apare de mai multe ori în orice ordine;

5) Etichetele pot fi numere şi/sau identificatori;

6) Constanta reală predefinită pi= 3.1415926536E+00;

7) Operatori suplimentari:
shl, shr (multiplicativi)
xor (aditivi)

8) Operatorul **not** poate fi aplicat unui operand întreg;

9) Instrucţiunea **case** cu clauza else;

10) Funcţii şi proceduri predefinite de lucru cu şirurile de caractere:

a) *delete* (**var** *s: string; p,n:byte);* şterge n caractere din şirul s începînd din poziţia p.

b) *insert (ss: string; var s:string; p:byte);* inserează subşirul ss în şirul s începînd din poziţia p a acestuia.

c) *str (v: integer; var s: string);* converteşte valoarea întreagă v într-un şir de caractere pe care-l depune în s;

d) *val (s:string;* **var** *v:tip;* **var** *cod: byte);* converteşte expresia şir de caractere s într-o valoare întreagă sau reală pe care o depune în v. Parametrul cod e pus pe 0 dacă nu apar erori sau indică poziţia primului caracter eronat;

e) *copy (s:string; p,n:byte): string;* crează un subşir format din n caractere ale şirului s, începînd din poziţia p a acestuia.

f) *concat (s1,s2,...,sn:string): string;* crează un şir prin concatenarea şirurilor s1,s2,...,sn. Acelaşi efect se obţine folosind operatorul +. Exemplu:
s:=s1+s2;

g) *length (s:string): byte;* dă lungimea şirului de caractere s.

h) *pos (ss,s:string):byte;* determină poziţia primei apariţii a subşirului ss în şirul s *(0 dacă nu apare).*

11) Acces la memorie şi la porturile de date folosinnd tablourile predefinite Mem şi Port.

12) Constante cu tipuri

a) nestructurate — sînt asemănătoare cu variabilele iniţializate. Exemplu:

```
const
  e:real=2.7182818;
  nume:string[7]=„POPESCU";
```

b) structurate — asigură tabluri şi mulţimi iniţializate (pentru conversii, teste, etc.).

0 **constantă tablou** iniţializează un tablou cu valori constante incluse între paranteze şi separate prin virgule. Exemplu:

```
type
  materii=(analiza, fizica, mecanica);
  sesiune=array [materii] of string[8];
const
  examen:sesiune=(„analiza", „fizica", „mecanica");
```

La definirea unei **constante înregistrare** se iniţializează cîmpurile înregistrării cu valori specificate. Exemplu:

```
type
  data=record
        zi  : 1...31;
        luna: 1...12;
        an  : 1900...2000
       end;
const
  azi:data=(zi:4; luna:9; an:1987);
```

La definirea unei **constante mulţime** se iniţializează elementele mulţimii cu valori dintr-o mulţime construită. Exemplu:

```
type
  litmari=set of 'A'..'Z';
const
  vocale: litmari=['A', 'E', 'I', 'O', 'U'];
```

13) proceduri predefinite de lucru cu fişiere

a) assign (VF, nume);
— asociază şirul de caractere „nume" cu variabila fişier VF.

b) rewrite (VF);
— crează un nou fişier identificat prin variabila fişier VF (modul generare)

c) reset (VF);
— pregăteşte în vederea prelucrării (în mod inspectare) fişierul identificat prin variabila fişier VF.

d) read (VF, Val);
— citeşte din fişierul disc identificat prin variabila fişier VF o componentă în variabila Val şi avansează poziţionarea la

următoarea componentă (modul inspectare).

e) write (VF, Val);

— scrie în fişierul disc identificat prin variabila fişier VF o componentă transferată din variabila Val şi avansează poziţionarea la următoarea componentă (modul generare).

f) seek (VF, n);

— avansează poziţionarea la cea de-a n-a componentă din fişierul identificat prin variabila fişier VF.

Pentru extinderea unui fişier după ultima sa componentă se utilizează: seek (VF, File Size (VF));

g) flush (VF);

— goleşte tamponul fişierului identificat prin VF realizînd astfel scrierea pe disc.

h) close (VF);

— închide fişierul disc asociat cu variabila fişier VF.

i) erase (VF);

— şterge fişierul disc asociat cu variabila fişier VF. Fişierul care se şterge trebuie în prealabil închis.

j) rename (VF, nume);

— fişierul disc asociat cu variabila fişier VF primeşte ca nume valoarea şirului de caractere nume. Fişierul trebuie în prealabil să fie închis.

k) EOF (VF);

— funcţie booleană ce furnizează valoarea „adevărat" dacă poziţionarea este la sfîrşitul fişierului (după ultima componentă).

l) FilePos (VF);

— funcţie întreagă ce furnizează poziţia curentă în fişier (prima componentă are poziţia 0).

m) Filesize (VF);

— funcţie întreagă ce dă numărul de componente ale fişierului (dimensiunea fişierului). Un fişier vid are 0 componente.

14) Controlul alocării memoriei la nivel de octet:

a) GetMem (VRef, N);

— aloca N octeţi în heap adresaţi prin variabila referinţa VRef.

b) FreeMem (VRef, N);

— eliberează N octeţi din heap care au fost alocaţi prin GetMem.

c) MaxAvail;

— furnizează dimensiunea celui mai mare bloc liber din heap.

15) Includerea de fişiere sursă în textul programului folosind directivă (•$I nume-fişier•).

16) Inserarea de instrucţiuni în cod maşină în textul programului prin instrucţiunea **inline.**

17) Alocarea de variabile la adrese absolute de memorie. Exemple:

```
var
  x: integer absolute $0000:$00F4;
```

în care primul parametru indică adresa segmentului bază iar al doilea deplasamentul; această formă este utilizabilă sub sistemele de operare PC-DOS, MS-DOS, CP/M-86. Sub CP/M-80 se indică numai adresa de memorie adică:

```
var
  x: integer absolute $40FA;
```

Două variabile pot fi **echivalate,** adică pot fi alocate începînd de la aceeaşi adresă de memorie. Exemplu:

```
var
  a: string[10];
  b: integer absolute a;
```

18) Obţinerea adresei din memorie a unui obiect (variabilă, procedură, funcţie) folosind funcţia Addr (nume).

19) Conversia tipurilor scalare folosind facilitatea Retype. Identificatorii de tipuri ordinale pot fi solosiţi ca desemnatori de funcţie cu un parametru. Exemple:

```
type
  zi=(lu, ma, mi, jo, vi, si, du);
  litere='a'..'z';
```

apelurile de mai jos întorc valorile:

```
integer (jo)=3
zi (5)=si
char (65)='A'
litere (3)='d'
```

20) Proceduri standard de intrare/ieşire pentru terminal

ClrEol — şterge toate caracterele din poziţia cursorului pînă la sfîrşitul liniei fără deplasarea cursorului.

ClrScr — şterge ecranul şi plasează cursorul în colţul stînga sus.

CrtInit — trimite pe ecran secvenţa de iniţializare a terminalului

CrtExit — trimite pe ecran secvenţa de resetare a terminalului, definită la instalare

Delay (Timp) — crează o buclă de întîrziere de ⟨Timp⟩ milisecunde

DelLine — şterge linia conţinînd cursorul, mutînd liniile de sub ea cu o poziţie în sus

InsLine — inserează o linie vidă în poziţia cursorului

GotoXY (X, Y) — mută cursorul în poziţia de pe ecran specificată prin valorile întregi ale celor 2 parametri (colţul stînga sus al ecranului are coordonatele 1,1).

Exit — produce ieşirea din blocul curent (întoarcere din subprogram sau terminarea programului)

Halt — opreşte execuţia programului cu întoarcere în sistemul de operare

LowVideo — setare ecran cu atributul 'Start of Low Video'

NormVideo — setare ecran cu atributul 'Start of Normal Video'

Randomize — Iniţializează generatorul de numere aleatoare
Move(S, D, N) — copiază un bloc de N octeţi din S la destinaţia D
FillChar (D, N, V) — umple o zonă de N octeţi începînd de la adresa D cu valoarea V.
Hi(I) — valoarea octetului superior a întregului I.
Lo(I) — valoarea octetului inferior a întregului I
KeyPressed — valoarea True dacă a fost acţionată o tastă de la consolă
Random — generează un număr aleator cuprins între 0 şi 1
Random (N) — generează un număr aleator cuprins între 0 şi N.
ParamCount — furnizează numărul de parametri transmişi programului în zona tampon a liniei comandă
ParamStr(N) — furnizează cel de-al N-lea parametru din zona tampon a liniei de comandă
SizeOf (Nume) — număr de octeţi ocupaţi în memorie de variabila sau tipul Nume
Swap(I) — interschimb între cei doi octeţi ai argumentului
UpCase (Car) — conversia în majusculă a argumentului

21) Facilităţi de segmentare a programelor.

Restricţii TURBO faţă de standard

1) Procedurile New şi Dispose nu acceptă înregistrări cu variante;
2) Procedurile standard Get şi Put nu sînt implementate;
3) Procedurile standard Pack şi Unpack nu sînt implementate. Sinbolul cuvînt packd nu are nici un efect (nu realizează o memorie mai economică);
4) Nu sînt permişi parametri procedurali;
5) Procedura Page nu este implementată;
6) Recursivitatea nu este o opţiune de compilare implicită;
7) Nu este implementat mecanismul tablourilor conforme.

Mesaje de eroare ale compilatorului

La apariţia unei erori în cursul compilării se afişează numărul erorii. Explicaţii suplimentare asupra erorii se dau numai dacă la lnasarea sistemului TURBO Pascal a fost încărcat fişierul cu mesaje de eroare TURBO.MSG. Lista mesajelor de eroare cuprinde:

01 ';' expected
02 ':' expected
03 ',' expected
03 '(' expected
05 ')' expected
06 '=' expected
07 ':=' expected
08 '[' expected
09 ']' expected
10 '.' expected
11 '..' expected
12 BEGIN expected
13 DO expected
14 END expected
15 OF expected
16 PROCEDURE or FUNCTION expected
17 THEN expected
18 TO or DOWNTO expected
20 Boolean expression expected
21 File variable expected
22 Integer constant expected
23 Integer expression expected
24 Integer variable expected
25 Integer or real constant expected
26 Integer or real expression expected
27 Integer or real variable expected
28 Pointer variable expected
29 Record variable expected
30 Simple type expected
31 Simple expression expected
32 String constant expected
33 String expression expected
34 String variable expected
35 Textfile expected
36 Type identifier expected
37 Untyped file expected
40 Undefined label
O instrucţiune se referă la o etichetă nedefinită
41 Unknown identifier or syntax error
În instrucţiune apare o entitate nedeclarată (o etichetă, un tip, o variabilă, un identificator de cîmp) sau o eroare de sintaxă
42 Undefined pointer type in preceding type definitions
Definirea unui tip pointer conţine o referinţă la un identificator de tip necunoscut
43 Duplicate identifier or label
Identificatorul sau eticheta a fost deja folosit în blocul curent
44 Type mismatch
1) tipul variabilei şi expresiei dintr-o atribuire sînt incompatibile
2) tipul parametrilor şi argumentelor dintr-o declarare şi un apel de procedură sau funcţie sînt incompatibili
3) tipurile operanzilor dintr-o expresie sînt incompatibile
45 Constant out of range (constanta depăşită)
46 Constant and CASE selector type does not match
Tipul selectorului şi constantelor etichete dintr-o instrucţiune CASE diferă
47 Operand type does not match operator
Tipul operanzilor nu corespunde operaţiei precizate (de exemplu: 'a' • 2)
48 Invalid result type
Tipul rezultatului (calculat de funcţie) este incorect.
Tipuri corecte sînt: tipurile scalare, tipurile şiruri de caractere (string), tipurile referinţă (pointer)
49 Invalid string length
Lungimea şirului de caractere nu este în intervalul 1...255

50 String constant lenght does not match type
51 Invalid subrange base type
Tipuri de bază pot fi numai tipurile scalare exceptînd tipul real.
52 Lower bound > Upper bound
Valoarea ordinală a limitei superioare trebuie să fie mai mare sau egală cu a limitei inferioare.
53 Reserved word
Cuvintele rezervate nu pot fi folosite ca identificatori.
54 Illegal assignment
55 String constant exceeds line
O constantă şir de caractere nu poate fi continuată de pe o linie pe alta.
56 Error in integer constant
O constantă întreagă nu se conformează regulilor de sintaxă sau depăşeşte domeniul —32768...32767.
57 Error in real constant
Constanta reală nu se conformează regulilor de sintaxă.
58 Illegal character in identifier
60 Constants are not allowed here.
61 Files and pointers are not allowed here.
62 Structured variables are not allowed here.
63 Textfiles are not allowed here.
64 Textfiles and untyped files are not allowed here.
65 Untyped files are not allowed here.
66 I/O not allowed here.
67 Files must be VAR parameters.
68 File components may not be files
Nu sînt permise construcţii de tipul file of file
69 Invalid ordering of fields
70 Set base type out of range
Tipul de bază al mulţimii trebuie să fie un scalar cu cel mult 256 valori posibile sau un subdomeniu cu limitele cuprinse între 0 şi 255.
71 Invalid GO TO
O instrucţiune GO TO nu poate referi o etichetă în interiorul unui ciclu FOR
72 Label not within current block
O instrucţiune GO TO nu poate referi o etichetă din afara blocului curent
73 Undefined forward procedure(s)
Un subprogram a fost declarat cu directiva forward, dar corpul lui nu apare nicăieri declarat.

74 INLINE error
75 Illegal use of ABSOLUTE
1) Într-o declaraţie de variabilă cu absolute, înaintea caracterului ':' poate apare numai un identificator.
2) Clauza absolute nu poate fi folosită într-o înregistrare
76 Overlays can not be forwarded
Directiva forward nu poate fi folosită în cazul segmentării
77 Overlays not allowed in direct mode
Segmentele pot fi folosite numai din programe compilate într-un fişier
90 File not found
Fişierul inclus specificat nu există
91 Unexpected end of source
Programul are probabil mai mulţi delimitatori begin decît end.
92 Unable to create overlay file
93 Invalid compiler directive
97 Too many nested WITHs
Folosiţi directivă de compilare W pentru a mări numărul instrucţiunilor WITH imbricate. În mod implicit (sub CP/M 80) sînt permise numai 2 instrucţiuni WITH imbricate.
98 Memory overflow
S-a încercat alocarea de mai multă memorie decît este disponibilă.
99 Compiler overflow
Memoria disponibilă este insuficientă pentru compilarea programului. Programul se împarte în mai multe segmente şi se folosesc fişiere incluse.

Diferenţe TURBO Pascal faţă de standard

Limbajul Turbo Pascal respectă limbajul standard Pascal definit de Wirth şi Jensen în „Pascal-User Manual and Repart", cu cîteva diferenţe minore apărute din motive de eficienţă a programelor executabile.

Variabile dinamice
Procedura New nu se poate folosi pentru înregistrări cu variante. Această restricţie poate fi ocolită prin utilizarea procedurii standard GetMem.

Recursivitate (in CP/M-80)
Din cauza modului în care sînt tratate variabilele locale într-un apel recursiv, o variabilă locală unui subprogram nu trebuie transmisă ca parametru 'var' într-un apel recursiv.

Get şi Put
Procedurile standard de I/E Get şi Put nu sînt implementate, dar procedurile Read şi Write au fost extinse pentru a satisface toate cerinţele operaţiilor de I/E. Motivele acestei decizii sînt trei: Read şi Write sînt mai rapide, memoria pentru variabile este mai mică deoarece nu se mai folosesc variabile buffer de fişier, iar procedurile Read şi Write sînt mai flexibile şi mai uşor de înţeles ca Get şi Put.

Instrucţiuni Goto
O instrucţiune goto nu poate face salt în afara blocului curent.

Procedura Page
Procedura standard Page nu este implementată, deoarece sistemul de operare CP/M nu defineşte un caracter de avans la pagina nouă.

Variabile împachetate
Cuvîntul rezervat 'packed' nu are nici un efect în Turbo-Pascal, dar este admis. Împachetarea se face automat oricînd este posibilă. Din acelaşi motiv nu sînt implementate procedurile standard Pack şi Unpack.

Parametri proceduri
Nu se pot transmite proceduri sau funcţii ca parametri la apelarea unor proceduri sau funcţii.

Mesaje de eroare la execuţie

Erorile apărute în cursul execuţiei produc oprirea programului şi afişarea unui mesaj de forma următoare:

Run-time error NN, PC=adr
Program aborted

unde NN este codul numeric al erorii, iar

'adr' este adresa de memorie unde s-a produs eroarea. Ambele numere sînt în baza 16.
01 Depăşire la calcule cu reali (în virgula mobilă)
02 Încercare de împărţire la zero
03 Argument negativ la funcţia sqrt de extragere radical
04 Argument negativ sau zero la funcţia de logaritmare Ln
10 Eroare de lungime şir de caractere:
1) Rezultatul concatenării a două şiruri are mai mult de 255 de caractere;
2) Conversie şir de lungime diferită de 1 în caracter
11 Indice incorect într-un şir
Expresie indice în afara domeniului 1...255 la apelarea procedurilor Copy, Delete, Insert.
90 Indice în afara limitelor
Expresie indice la un element de tablou în afara domeniului declarat pentru acel tablou
92 Întreg prea mare
La conversie din real în întreg cu Trunc sau Round rezultă o valoare în afara domeniului —32768...32767
F0 Fişier cu segment de program (overlay) negăsit
FF Memorie insuficientă la alocare dinamică.
Apel la procedura New sau procedura recursivă şi nu mai există memorie liberă între memoria 'heap' (adresa HeapPtr) şi vîrful stivei de recursivitate (adresa RecurPtr)

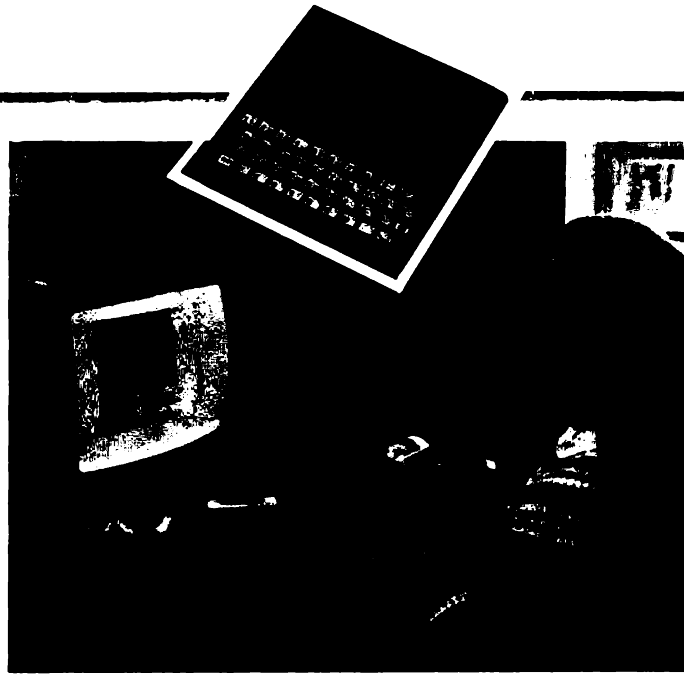
Mesaje de eroare la intrări—ieşiri

Dacă se produce o eroare la operaţii de intrare—ieşire şi este activă directiva I de verificare automată a rezultatului I/E atunci programul se opreşte şi se afişează următorul mesaj de eroare:

I/O error NN, PC=adr
Program aborted

unde NN este codul numeric al erorii de I/E, iar 'adr' este adresa de memorie unde s-a produs eroarea (ambele în hexazecimal). Dacă este dezactivată verificarea operaţiilor de I/E, {$I-}, atunci programul nu se opreşte, dar nu mai trebuie executate alte operaţii de I/E pînă cînd nu se apelează funcţia IOresult pentru examinarea rezultatului operaţiei de I/E; în caz contrar programul se poate 'pierde' într-o altă eroare de I/E.

01 *Fişier inexistent*
Nu există un fişier cu numele dat în procedurile Reset, Erase, Rename, Execute, Chain.
02 *Citire din fişier ne-deschis*
1) Încercare de citire (cu Read sau Readln) dintr-un fişier care nu a fost deschis anterior prin Reset sau Rewrite
2) Încercare de citire dintr-un fişier text deschis prin Rewrite şi deci gol
3) Încercare de citire de la dispozitivul LST:, care se poate folosi numai pentru scriere.
03 *Scriere în fişier ne-deschis*
1) Încercare de scriere (cu Write sau Writeln) într-un fişier care nu a fost deschis prin Reset sau Rewrite,
2) Încercare de scriere într-un fişier text deschis cu Reset,
3) Încercare de scriere la dispozitivul logic KBD:, care se poate folosi doar pentru citire.
04 *Fişier ne-deschis*
Încercare de acces (cu BlockRead sau BlockWrite) la un fişier, fără un Reset sau Rewrite anterior pe acel fişier.
10 *Eroare de format la citire numere*
Şirul citit dintr-un fişier text într-o variabilă numerică nu respectă formatul pentru contante numeerice
20 *Operaţie interzisă la un dispozitiv logic*
Încercare de utilizare proceduri Erase, Rename, Execute sau Chain pentru un fişier asociat unui dispozitiv logic.
21 *Operaţie interzisă în modul direct*
Încercare de apelare Execute sau Chain dintr-un program executat în mod direct, adică prin comanda Run din Turbo cu opţiune de compilare în memorie.
22 *Atribuire la fişiere standard prin Assign*
90 *Lungime de înregistrare incorectă*
Lungimea înregistrării la o variabilă fişier diferă de lungimea efectivă a înregistrărilor din fişierul extern cu care se asociază variabila.
91 *Poziţionare prin Seek dincolo de sfîrşitul fişierului*
99 *Sfîrşit de fişier neaşteptat*
1) La citirea dintr-un fişier text nu s-a găsit caracterul Ctrl-Z şi s-a ajuns la sfîrşitul fizic al fişierului
2) Încercare de citire după sfîrşitul unui fişier
3) Nu se poate citi sectorul următor din fişier cu Read sau cu BlockRead.
F0 *Eroare la scriere pe disc*
Disc plin la încercarea de extindere a unui fişier. Această eroare poate apare la scriere (Write, BlockWrite, Flush) dar şi la închidere de fişier sau la o citire precedată de golirea zonei tampon pe disc.
F1 *Tabela directoare plină*
Nu mai este loc în tabela directoare a discului la crearea unui nou fişier cu Rewrite (prea multe fişiere).
F2 *Fişier prea mare*
Încercare de scriere a unei înregistrări cu număr mai mare de 65535.
F3 *Prea multe fişiere deschise*
FF *Fişier dispărut*
Încercare de închidere, prin Close, a unui fişier negăsit în tabela directoare, de exemplu datorită schimbării discului.



Ion DIAMANDI
Florin VASILIŢĂ

Utilizarea rutinei de citire din memoria - ROM

Informaţiile referitoare la un fişier salvat pe caseta magnetică (tipul fişierului, lungimea sa etc.) se regăsesc în partea de antet („header“) care însoţeşte fişierul. Într-adevăr, fişierele sînt separate în două părţi, fiecare avînd un format diferit. Prima parte sau antetul fişierului conţine diverse informaţii relative la fişier. Ea ocupă 17 octeţi organizaţi în felul următor:

Partea a doua a fişierului conţine datele propriu-zise ale fişierului. Pentru un fişier care memorează un program în cod maşină, această parte va conţine toţi octeţii programului.

Interfaţa cu casetofonul. Bitul 3 al portului 254 este utilizat pentru scriere **(OUT)** iar bitul 6 este utilizat pentru citire **(IN).** Aceşti doi biţi sînt înlocuiţi de rutinele de scriere şi citire casetă care sînt situate în memoria ROM şi care permit scrierea sau citirea de pe casetă a unui număr de octeţi. Numărul de octeţi, precum şi poziţia sînt fixate de utilizator.

Rutina de scriere este situată la adresa 4C2H (1218). La început, **IX** va trebui să conţină adresa de început a zonei de transferat, iar **DE,** numărul de octeţi care se transferă. Registrul **A** va avea valoarea **0** (dacă se doreşte scrierea unui antet de fişier) sau **FFH** (dacă se doreşte scrierea corpului unui fişier). Următoarea suită de instrucţiuni va permite scrierea uneia din părţile fişierului pe caseta magnetică:

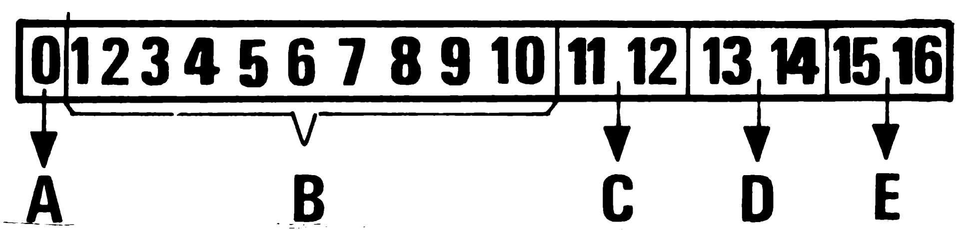
R1

Rutina de citire este situată la adresa **556H (1366)** şi necesită aceleaşi valori de intrare. În plus, va fi necesar ca indicatorul **Carry** să fie poziţionat pe 1. Următoarea suită de instrucţiuni va permite citirea de pe casetă a unei părţi a fişierului:

R2

Iată şi suita de instrucţiuni (dezasamblate cu MONS) care permit citirea informaţiilor din antet referitoare la fişier (pe prima coloană, adresele instrucţiunilor în zecimal; a 2-a coloană, codurile instrucţiunilor în hexazecimal; a 3-a coloană, mnemonicele de instrucţiuni cu operanzi în zecimal; a 4-a coloană, comentarii):

R3



A.Tipul fişierului: 0=program BASIC („program“); 1=fişier de date numerice („number array“); 2=fişier de date alfanumerice („character array“); 3=program cod maşină („bytes“); 4=program cod maşină creat prin Editor/Asamblor. **B.Numele fişierului. C.Lungimea fişierului salvat:** - lungimea programului în octeţi împreună cu variabilele (pentru tipul Ø-program BASIC);-lungimea programului sau fişierului cod maşină (pentru tipul 1, 2 sau 3). **D.Linia de autostartare** (pentru tipul 0-program BASIC). **Adresa de implantare a programului** (pentru tipul 3-program în cod maşină). (Dacă tipul este 1 sau 2-fişier de date-nu se pune nimic.) **E.Lungimea programului BASIC fără variabile.** (Dacă tipul este 1, 2 sau 3 nu se pune nimic.)

R1

```
LD      IX , adresa     ; adresa de inceput a zonei
                        ; care contine octetii de transferat
LD      DE , numar      ; numarul de octeti de transferat
LD      A  , cod        ; 00 sau FFH in functie de partea de trasferat
                        ; (antet sau fisier propriu-zis)
CALL    4C2H            ; scriere
```

R2

```
LD      IX , adresa     ; adresa zonei in care vor fi transferati
                        ; octetii cititi
LD      DE , numar      ; numarul de octeti de citit
LD      A  , cod        ; 00 sau FFH in functie de partea de citit
                        ; (antet sau fisier propriu-zis)
SCF                     ;
CALL    556H            ; citire
```

R3

```
27986 37          SCF             ; set carry flag
27987 3E00        LD A , 00       ; se incarca acumulatorul cu 0
27989 DD21606D    LD IX , 27000   ; se incarca IX cu adresa zonei
                                  ; in care vor fi transferati octetii
                                  ; (28000)
27993 111100      LD DE , 00017   ; se incarca DE cu 17 (numarul de
                                  ; octeti ai antetului)
27996 CD5605      CALL 01366      ; se apeleaza rutina de citire din ROM
27999 C9          RET
```

Următorul program reprezintă o aplicaţie care va utiliza **informaţiile extrase din antetele fişierelor** existente pe caseta magnetică **prin intermediul rutinei de citire casetă** prezentate anterior. Programul va crea un fişier de date pentru ţinerea evidenţelor referitoare la programele conţinute într-o bibliotecă personală.

Rutina de citire a informaţiilor din antet se va localiza la adresa **27986** (vezi linia 1) şi va fi apelată din linia 150.

Rutina de citire se introduce din **BASIC** prin intermediul unei linii **DATA** (linia 30) care conţine codificarea instrucţiunilor în cod maşină şi a operanzilor în zecimal. Se poate observa că: 55 = 37 H; 62 = 3EH etc., regăsindu-se astfel codurile instrucţiunilor în hexazecimal din programul dezasamblat (coloana a 2-a). Datele din linia **DATA** au fost introduse ca şiruri de caractere, urmînd a fi transformate, apoi, în program în valori numerice prin intermediul funcţiei **VAL.** În acest mod se realizează o economie de memorie.

În linia 50 se realizează o rezervare de memorie începînd de la adresa **28 000** (adică de unde se termică programul în cod maşină) şi de la această adresă se va memora fişierul de date cu informaţiile conţinute în antetele citite.

Utilizare. Programul prezentat este un program utilitar cu ajutorul căruia se poate ţine evidenţa înregistrărilor pe casete magnetice. Memorarea informaţiilor (evidenţelor) se va face tot pe caseta magnetică.

Programul prezintă un tablou pe care vor fi afişate informaţiile. După lansarea în execuţie a programului se porneşte casetofonul pentru parcurgerea casetei înregistrate cu programe. Programul va afişa (eventual pe mai multe pagini — ecrane — care conţin cîte 9 titluri) denumirea fişierului, **tipul** (program BASIC — **Prog** —, program cod maşină — **Bytes** — fişier de date numerice — **Num A** — sau fişier de date alfanumerice — **Chr A**), **lungimea fişierului şi linia de autostartare** (pentru programe BASIC) sau **adresa de implantare a programului** (pentru programe în cod maşină).

La terminarea parcurgerii unei casete sau cînd utilizatorul doreşte se va acţiona orice tastă (dar numai pînă la 4-5 secunde de la afişarea informaţiilor referitoare la fişierul citit) şi se va intra în al doilea mod de lucru, ale cărei opţiuni sînt: S (Sus) pentru afişarea paginii inferioare, V (Vîrf), pentru afişarea primei pagini, J (Jos) pentru afişarea paginii superioare, R (Reîntoarcere) — pentru intrare în primul mod de lucru, T (Tipărire) — pentru imprima-

rea fişierului la imprimantă. In cazul tastării lui R (Reîntoarcere) apar alte 3 opţiuni: S (Salvare) pentru salvarea programului împreună cu fişierul creat, N (Nou) — pentru ştergerea fişierului creat şi trecerea la o nouă casetă, C (Continuare) — pentru explorarea în continuare a casetei.

În cazul testării lui T (Tipărire) apare două opţiuni şi anume: I (Întreg) — pentru tipărirea la imprimantă a tuturor paginilor fişierului (întreg fişier) şi P (Pagină) — pentru tipărirea numai a paginii curente.

Se recomandă ca fişierul creat să se salveze la începutul casetei respective.

Menţionăm faptul că programul decodifică numai informaţiile din antete. Dacă în cazul unor fişiere antetul lipseşte (aceste fişiere sînt denumite de unele programe de copiere „headerless") atunci aceste fişiere vor fi ignorate de programul utilitar. Introducerea valorii 5 la adresa 28 000 (linia 1), care reprezintă începutul fişierului creat, şi, apoi (eventual) la adresa 28 000 + multiplu de (17+1) se realizează în program tocmai în scopul ignorării fişierelor care nu sînt de tipul 0, 1, 2, 3 sau 4.

La o ieşire accidentală programul se poate relua cu CONTINUE sau cu RUN, caz care este echivalent cu opţiunea N pentru casetă nouă.

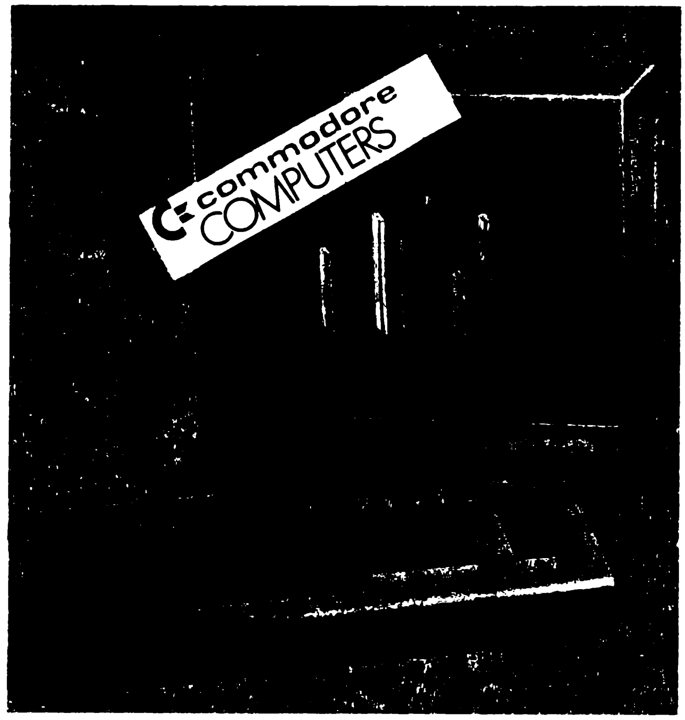
```
  1 CLEAR 27985: POKE 28000,5
  2 LET c8=9: LET p=1: LET c=1
  8 INPUT "Cod Caseta (2 cifre)
:";y
 10 BORDER 0: PAPER 0: INK 7: C
LS
 20 DATA "55","62","0","221","3
3","96","109","17","17","0","205
","86","5","201"
 30 GO SUB 40: GO TO 80
 40 RESTORE 20: FOR a=27986 TO
27999: READ a$: POKE a,VAL a$: N
EXT a
 50 DEF FN p(a)=PEEK a+256*PEEK
 (a+1)
 60 DEF FN a(p,c)=28000+17*(c-1
)+17*c8*(p-1): RETURN
 70 INPUT :: PRINT #1: PAPER 1;
 FLASH 1;"     Opriti casetofonul
   "'; FLASH 0;TAB 25;"PAG ";p:
RETURN
 80 GO SUB 500
150 RANDOMIZE USR 27986: GO SUB
 780: IF PEEK FN a(p,c)=5 THEN G
O TO 150
160 GO SUB 600
170 GO SUB 800
180 IF c<c8 THEN LET c=c+1: LET
 et=150: GO TO 200
190 IF c=c8 THEN LET p=p+1: LET
 c=1: LET et=80: GO TO 200
200 POKE FN a(p,c),5: POKE 2799
2,INT (FN a(p,c)/256): POKE 2799
1,FN a(p,c)-256*INT (FN a(p,c)/2
56)
210 GO TO et
500 REM Desen cap de tabel
520 CLS : PRINT PAPER 1;"DIR*Ca
seta ";y;" Adr.Lungime.Auto"''
530 PLOT 0,165: DRAW INK 2;255,
0
540 PLOT 0,166: DRAW INK 2;255,
0
550 INPUT :: PRINT #1: PAPER 1;
 FLASH 1;"Lasati casetofonul por
nit!"'; FLASH 0;TAB 25;"PAG ";p:
 RETURN
570 LPRINT "DIR*Caseta ";y;" Ad
r.Lungime.Auto"
580 FOR i=0 TO 31: LPRINT "-";:
 NEXT i: LPRINT : RETURN
600 REM Decodificare heder
602 GO SUB 610: GO SUB 680: RET
URN
610 LET ad=FN a(p,c)
620 LET t=PEEK ad
630 DIM n$(10)
640 FOR i=1 TO 10: LET n$(i)=CH
R$ PEEK (ad+i): NEXT i
650 LET l=FN p(ad+11)
660 LET s=FN p(ad+13)
670 LET x=FN p(ad+15)
672 RETURN
680 GO SUB 700+t: GO TO 710
700 PRINT "Prog. ";: RETURN
701 PRINT "Num.A ";: RETURN
702 PRINT "Chr.A ";: RETURN
703 PRINT "Bytes ";: RETURN
705 RETURN
710 PRINT PAPER 1;n$; PAPER 0;"
";
720 IF t=3 THEN PRINT s;",";l;:
 GO TO 770
730 PRINT l;
740 IF t>0 THEN GO TO 770
750 PRINT "/";x;
760 IF s>=0 AND s<10000 THEN PR
INT "/";s;
770 PRINT '': RETURN
780 BEEP .1,-16: BEEP .1,32: BE
```

```
EP .1.31: BEEP .1.48: RETURN
 805 INPUT ;: PRINT #1: FLASH 1:
 PAPER 2;"Pentru OPRIRE actionat
i o TASTA    URGENT !  ": FLASH
0: PAPER 1;TAB 25;"PAG ";p
 810 FOR w=1 TO 180
 820 IF INKEY$="" THEN GO TO 100
0
 830 LET a$=INKEY$: GO SUB 780
 840 LET cc=c: LET pp=p
 850 INPUT ;: PRINT #1; FLASH 1;
" ": FLASH 0: PAPER 1;"SusVirfJo
s ReitoarcereTiparire";TAB 25;"P
AG ";p
 860 IF INKEY$="" THEN GO TO 860
 870 LET a$=INKEY$: GO SUB 780
 880 IF a$="r" THEN LET p=pp: GO
 SUB 500: GO SUB 70: FOR c=1 TO
cc: GO SUB 600: NEXT c: LET c=cc
: INPUT "salvare sau noua caseta
 sau";TAB 2;"continuare?";a$:: I
F a$="s" THEN GO SUB 780: GO SUB
 9999
 881 IF a$="n" THEN GO SUB 780:
RUN
 884 IF a$="t" THEN : GO TO 2500
 890 IF a$="s" THEN LET p=p-1: G
O TO 920
 900 IF a$="v" THEN LET p=1: GO
TO 920
 904 IF a$="c" THEN GO TO 914
 910 IF a$="j" THEN LET p=((p AN
D p=pp)+((p+1) AND p<pp))
 912 GO TO 920
 914 LET p=pp: GO SUB 950: LET c
=cc: GO TO 1030
 920 IF p<1 THEN LET p=1
 930 IF p>pp THEN LET p=pp
 940 GO SUB 500: GO SUB 70: GO S
UB 960: GO TO 850
 950 GO SUB 500
 960 FOR c=1 TO ((c8 AND p<pp)+(
cc AND p=pp))
 970 GO SUB 600
 980 NEXT c
 990 RETURN
1000 NEXT w
1010 GO SUB 550
1020 RETURN
1030 LET w=w+1: IF w<179 THEN GO
 TO 1030
1040 GO TO 1000
1100 REM Tiparire la imprimanta
1110 GO SUB 610
1120 GO SUB 1200+t
1130 GO TO 1210
1200 LPRINT "Prog. ";: RETURN
1201 LPRINT "Num.A ";: RETURN
1202 LPRINT "Chr.A ";: RETURN
1203 LPRINT "Bytes ";: RETURN
1205 RETURN
1210 LPRINT n$;" ";
1220 IF t=3 THEN LPRINT s;".";l;
: GO TO 1270
1230 LPRINT l;
1240 IF t>0 THEN GO TO 1270
1250 LPRINT "/";x;
1260 IF s>=0 AND s<10000 THEN LP
RINT "/";s;
1270 LPRINT : RETURN
2500 INPUT #1;"Tiparire intreg f
isierul sau    numai pagina cure
nta?";a$
2510 IF a$="p" THEN GO SUB 780:
GO SUB 2600
2520 IF a$="i" THEN GO SUB 780:
GO SUB 2700
2530 GO TO 850
2600 GO SUB 570: FOR c=1 TO ((c8
 AND p<pp)+(cc AND p=pp)): GO SU
B 1100: NEXT c
2610 LPRINT ';TAB 25;"PAG.";p: G
O SUB 580: LPRINT '''
2620 LET c=((c8 AND p<pp)+(cc AN
D p=pp)): RETURN
2700 FOR p=1 TO pp: GO SUB 2600:
 NEXT p: LET p=pp: RETURN
9000 BORDER 0: PAPER 0: INK 7: C
LS : PRINT AT 10,2; PAPER 2; FLA
SH 1;"Fisierul Director se incar
ca": GO SUB 550: LOAD ""CODE
9010 GO SUB 950: GO SUB 40: GO T
O 180
9999 LET y$="DIR*Cas "+STR$ y: S
AVE y$ LINE 9000: SAVE "director
"CODE 28000,17*c+17*c8*(p-1): RE
TURN
```

```
DIR*Caseta 93 Adr.Lungime.Auto
-------------------------------
Prog. ManicMiner 94/94/1
Bytes mm1         22528.768
Bytes MM2        32768.32768
Prog. Android 1   977/977/200
Bytes M/C        24960.7508
Bytes Screen      16384.6912
Prog. CASETA PR1 63/63/1
Bytes CASETA SR1 16384.6912
Prog. CASETA PR1 546/467/10
```

Călin OBRETIN

INFORMATICA -un joc serios !



Toţi cei ce folosesc calculatorul în timpul liber cunosc cît de dificil este să ajungi la sfîrşitul unui program, mai ales cînd acesta este de tipul „trage înaintea adversarului".

Cred că este util să ştim cîteva POKE-uri care să ne ajute să urmărim un joc pînă la capăt. Aceste instrucţiuni se introduc după ce s-a încărcat programul de pe bandă sau disc; după această operaţie se tastează comanda RUN.

JOCUL	POKE	EFECT
Quest of tires	7341,199	vieţi infinite
Donkey Kong	12118,234	idem
Sea fox	7337,173	idem
Shamus I	18486,162 23558,169	idem
Shamus II	15475,238	idem
Zaxxon	11353,X	X-nr. de vieţi
Lady TUT	2392,5	vieţi infinite

Tot pentru posesorii de **Commodore 64** indicăm şi alte instrucţiuni utile ce se pot folosi la crearea de programe:

SYS 65499 — resetează ceasul (intern) la 00:00;

SYS 42115 — cursorul funcţionează fără READY;

SYS 44808 — apare mesajul ?SYNTAX ERROR;

WAIT 653,1..2..4 — calculatorul aşteaptă apăsarea tastelor SHIFT;C =;, respectiv CTRL;

WAIT 203,63 — se aşteaptă apăsarea tastei cu codul 63.

Puţini cunosc că un **Commodore 64** poate funcţiona în două moduri:

SLOW

POKE 53265,PEEK-(53265) AND 239 : POKE 53296,PEEK-(53296) OR 1

FAST

POKE 53296,PEEK-(53296) AND 254 : POKE 53265,PEEK-(53265) OR 16

Tot în mod FAST pot lucra şi cei ce au un **Commodore 128** prin simpla comandă:

POKE 53424, PEEK(53424) OR 3

Şi pentru a încheia această scurtă listă, cîteva facilităţi ale microprocesorului 6510 la scrierea programelor:

- programe BASIC alocate în zona C000
  POKE 44,192
  POKE 56,208 :
  POKE 49152,0 : NEW
- dispariţia cursorului
  SYS 64328 (comanda se restabileşte cu tastele RUN/STOP-RESTORE)
- cursor rapid
  POKE 6325,5...255
- dezactivare (activare) tasta RUN/STOP
  POKE 788,52...49
- dezactivare (activare) tasta RESTORE
  POKE 792,193...71
- cursor static 4 sec.
  POKE 651,255.

Şi o ultimă instrucţiune: cînd vrem să citim un fişier de pe bandă, iar această citire se face în mijlocul unui alt program ce apelează acest fişier (deci citirea se face în timpul rulării unei secvenţe BASIC) se foloseşte:

POKE 3600,266 :SYS7082 : LOAD

Această comandă are ca efect rularea în continuare a secvenţei BASIC fără întreruperea ei, fişierul dorit fiind în memoria calculatorului.

Virgil IONESCU
Radu CONSTANTINESCU

Interfață serială pentru calculatoare compatibile HC-85

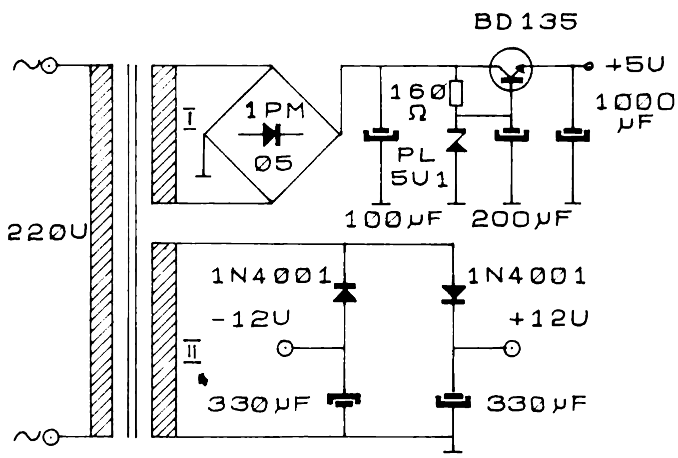
Rezultatele unor programe de calcul, graficele obținute în urma prelucrării unor date, fișierele create cu editoare de text, listingurile unor programe lungi sînt greu de urmărit numai pe display. O imprimantă cuplată la calculator poate ușura foarte mult munca. Deoarece HC-85 nu are prevăzută constructiv o interfață care să permită transmisia datelor către o imprimantă, prezentăm în continuare un astfel de dispozitiv. Principalele sale caracteristici sînt:

Tip: **serială, nestandard**
Compatibilitate: **RS232C**
Protocol: **DTR (Data Terminal Ready)**
Viteza de transfer: **9600 bauds**
Formatul transmisiei:
1 bit de start;
8 biți de date, fără paritate;
1 bit de stop.

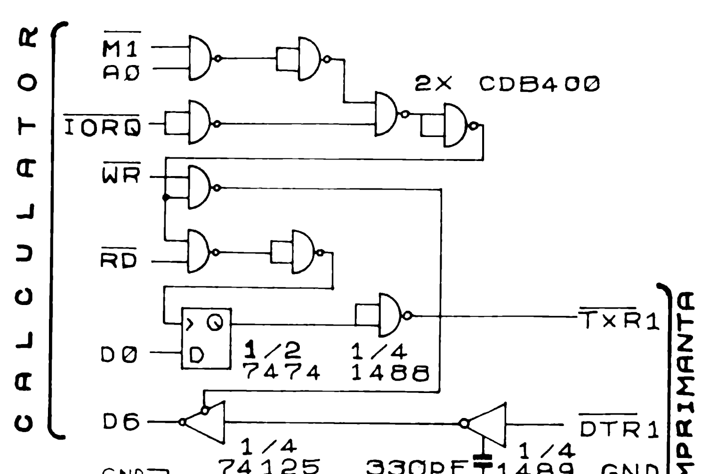
Utilizarea protocolului DTR permite cuplarea interfeței cu imprimanta numai prin trei linii (masă, linia de transmisie a datelor, linia de recepție a semnalului DTR). Transmisia datelor (TxD) se face prin intermediul bitului D1 al portului de adresă 1 iar recepția semnalului DTR folosește bitul D5 al aceluiași port.

Partea hardware (fig. 1), utilizînd circuite nespecializate, accesibile, a fost concepută pentru o cît mai mare simplitate, folosind un minimum de componente. Decodificarea liniilor de comandă (IORQ, M1, WR, RD) și a liniei AO se face cu porți NAND (CDB 400). Pentru ieșirea datelor este utilizat un bistabil de tip D (7474), iar pentru intrare o poartă de mare impedanță (74125). Adaptarea la nivelurile cerute de standardul RS232C se realizează pe fiecare linie, cu cîte un translator de nivel (1488, 1489). Cuplarea cu calculatorul este asigurată de conectorul interfață (fig. 2), iar cu imprimanta, de un conector cu cel puțin 3 contacte. Corespondența dintre ieșirile interfeței și intrările imprimante este următoarea:

Interfață...Imprimantă
TxR1...RxD
DTR1...DTR
GND...GND

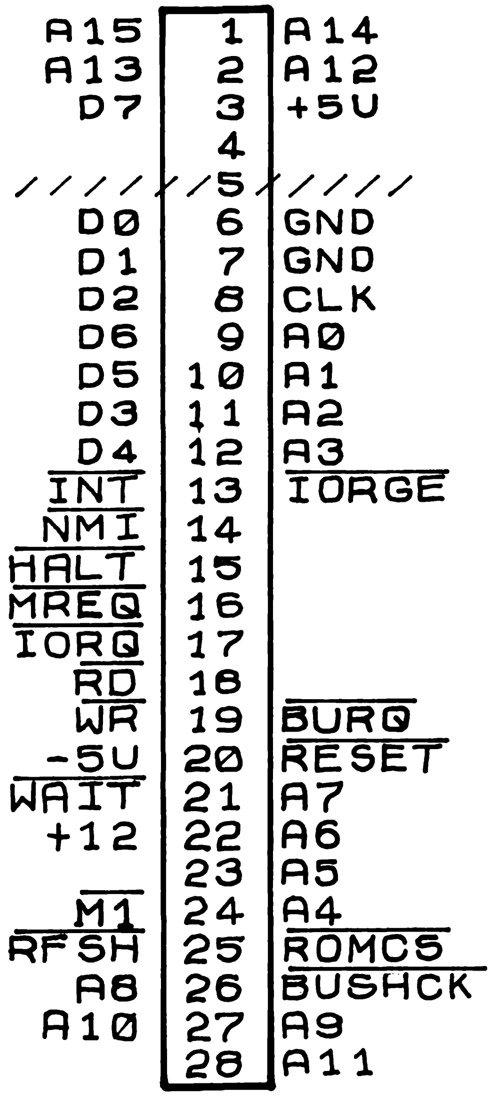


3 Alimentatorul



1 Interfața

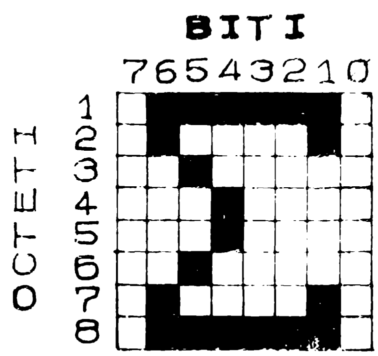
▲ Atelier Spectrum Sinclair

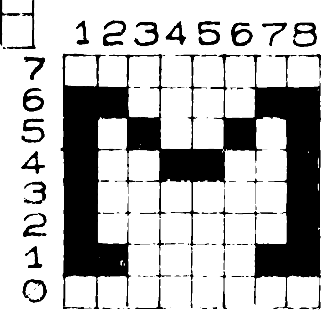


❷ Conectorul interfață

Alimentarea poate fi asigurată și din sursa calculatorului, dar, deoarece aceasta este proiectată aproape la limită, este preferabil să se utilizeze un alimentator separat capabil să furnizeze cele trei tensiuni necesare. În figura 3 este prezentată schema unui astfel de alimentator. Transformatorul se obține dintr-unul de sonerie, rebobinîndu-i secundarul pentru realizarea celor două înfășurări: I=156 spire CuEm diametru 0,5 și II=200 spire CuEm diametru 0,2.

```
LP      CALL  #1F54  BREAK
        JP    C,PR0
        RST   #08
        DEFB  14
PR0     IN    A,(1)
        BIT   5,A
        JR    NZ,PRINT
        DI
        XOR   A
        CALL  PR_OUT
        LD    A,B
        LD    B,8
PR1     CALL  PR_OUT
        RRA
        DJNZ  PR1
        LD    A,1
        CALL  PR_OUT
        EI
        RET
PR_OUT  OUT   (1),A
        PUSH  BC
        LD    B,21
        DJNZ  TIMER
        POP   BC
        RET
```

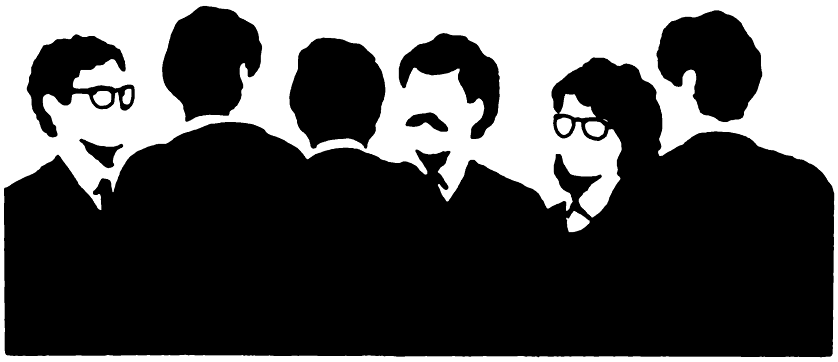




Soft-ul de comandă conține ca principală rutină pe cea de transmitere a unui octet la imprimantă (listing 1). Este prevăzută posibilitatea de a comanda BREAK înainte de recepționarea fiecărui octet. Bucla de întîrziere este necesară pentru a realiza adaptarea între viteza de transmisie și cea de recepție. Valoarea din listing (21) corespunde unei viteze de 9 600 bauds și poziționării rutinei în RAM-ul superior.

Autorul pune la dispoziția doritorilor programe care exploatează facilitățile imprimantei matriciale ROBOTRON K6313-seriale, folosibile și pentru alte imprimante respectînd particularitățile acestora.

Promitem să revenim cu aplicații interesante: modificarea programului Tasword II pentru LPRINT și COPYCOLOR — un program de transpunere în tonuri de gri a culorilor ecranului, sau cu alte sugestii pe care le așteptăm din partea dv.



Liviu DUMITRAŞCU
Stelian GUŢU

ADMITEREA – produs software educaţional pentru pregătirea candidaţilor la concursul de admitere în învăţămîntul superior

Importanţa calculatorului în procesul de învăţămînt este un fapt de netăgăduit.
Argumentele pro sînt numeroase şi ele au făcut deja obiectul a numeroase anchete, calculatorul fiind în prezent în centrul atenţiei factorilor de decizie pentru a soluţiona optim această problemă. Desigur, calculatorul poate interveni în numeroase faze ale procesului de învăţămînt, atît în informatizarea învăţămîntului, cît şi în informatica tratată ca obiect de studiu.
În cele ce urmează, vă propunem o foarte sumară prezentare a unui pachet de programe (în dBASE) dedicate unui subiect foarte „fierbinte": admiterea în învăţămîntul superior.

Scop: Pregătirea asistată de calculator a examenului de admitere în învăţămîntul superior la disciplina „fizică". Se utilizează microcalculatoare de tip M118, TPD, JUNIOR, CUB-Z sub sistemul de operare CP/M.

Cui se adresează: Produsul poate fi utilizat de către candidaţii la concursul de admitere în învăţămîntul superior la disciplina „fizică", cadrele didactice din învăţămîntul liceal, elevii care doresc să se pregătească în vederea acestui examen.

Scurtă descriere: „Admitere" este un sistem de programe care gestionează o bază de date semnificativă din punct de vedere al conţinutului, cuprinzînd toate problemele şi subiectele de teorie date la admitere în învăţămîntul superior, disciplina „fizică" în ultimii zece ani în ţară. Interogarea bazei de date poate fi efectuată uşor, simplu şi comod de

către utilizatorul obişnuit, fără o pregătire informatică prealabilă. Punctul de plecare îl constituie alegerea tipului de învăţămînt: învăţămîntul tehnic, învăţămîntul universitar (profilul fizică), învăţămîntul medico-farmaceutic (medicină şi stomatologie). Învăţămîntul tehnic este structurat pe următoarele profiluri:

T1: mecanic, electric, energetic, mine, petrol, metalurgie, construcţii, geodezie, tehnologie şi chimia textilelor, tehnologia şi chimia produselor alimentare, forestier (specializare industria lemnului), geologie.

T2: chimie şi mecano-chimic.

T3: învăţămîntul de subingineri (trei ani) — toate profilurile, specializările din învăţămîntul pedagogic de chimie-fizică (trei ani) şi matematică-fizică (trei ani).

Învăţămîntul medico-farmaceutic conţine profilurile medicină şi stomatologie. Mai departe, problemele aparţinînd profilurilor menţionate sînt structurate pe capitole: mecanică, fizică moleculară şi căldură, electricitate, optică (numai pentu profilurile medicină şi stomatologie, precum şi învăţămîntul universitar, profilul fizică), fizică atomică şi nucleară (pentru învăţămîntul universitar, profilul fizică). Fiecare capitol este, la rîndul său, structurat pe clase de probleme. De exemplu, capitolul „electricitate“, comun tuturor candidaţilor, cuprinde următoarele clase de probleme:

1. Probleme de electrostatică cu: cîmp electric şi potenţial electric; condensatoare şi energia cîmpului electric dintre armături; mişcarea particulelor în cîmp electric.
2. Probleme de electrocinetică cu: legea lui Ohm şi teoremele lui Kirchoff; bilanţul energetic; legile electrolizei.
3. Probleme de magnetism: de cîmp magnetic al curentului electric; de mişcare a particulelor în cîmp electric şi magnetic; de inducţie electromagnetică.
4. Probleme cu circuite în curent alternativ.
5. Probleme cu tuburi electronice şi semiconductoare.
6. Probleme combinate.

Fiecărei probleme îi este ataşat un grad de dificultate de la 1-3 corespunzător următoarelor clasificări:

Gradul 1: probleme relativ simple care se rezolvă aplicînd, de regulă, formula (majoritatea problemelor date la examenele de subingineri);

Gradul 2: probleme mai complexe care se rezolvă aplicînd o anumită succesiune de formule;

Gradul 3: probleme la care înainte de a aplica formulele cunoscute, trebuie efectuate anumite artificii (de calcul, înţelegerea şi interpretarea fenomenelor etc.).

Mod de utilizare: Utilizatorul are posibilitatea să selecteze din tezaurul aflat la dispoziţia sa toate problemele aparţinînd unui interval sau pe ultimii n ani (n cuprins între 1-10) specificînd capitolul (profilul, tipul de învăţămînt) şi gradul de dificultate. După ce va încerca rezolvarea, utilizatorul va putea confrunta rezultatul cu răspunsul oferit de sistem. Menţionăm că, începînd cu anul 1988, problemele conţin şi baremurile cu punctajul fixat de minister.

Avantaje: Tezaur complet pe ultimii zece ani (teorie plus probleme); acoperă toate profilurile; grad de detaliere dorit de utilizator; grad de dificultate adaptat nivelului de pregătire al utilizatorului; răspunsuri exacte garantate de sistem.

Mod de livrare. Programele şi baza de date se află pe suport magnetic structurate pe trei domenii: învăţămîntul tehnic, învăţămîntul universitar (profilul fizică) şi învăţămîntul medico-farmaceutic. Baza de date (pe ultimii zece ani) conţine atît subiectele de teorie (fără rezolvări) cît şi problemele propriu-zise cu soluţiile lor.

Dischetele sînt însoţite de un manual care cuprinde: descrierea produsului, modul de utilizare şi programele-sursă.

Redacţia noastră intenţionează să publice un supliment dedicat acestei baze de date, în ipoteza în care cerinţa va fi suficient de mare!

Aşadar, după ce veţi lua cunoştinţă despre această inedită bază de date, extrem de necesară tuturor candidaţilor (noi o oferim „în lucru“ pentru anul şcolar 1990-1991), aşteptăm răspunsul dv. făcînd menţiunea pe plic: „ADMITERE“.

Vă mulţumim!

Art.1 — Clubul Român de Calculatoare este o organizaţie apolitică, reunind prin afiliere liberă atît persoanele ce posedă un calculator personal, cît şi cei ce doresc iniţierea în acest domeniu.

Art.2 — Condiţia unică de apartenenţă la Clubul Român de Calculatoare o constituie aprobarea statului organizaţiei. (...)

Art.7 — Principalul obiectiv al CRC este liberul schimb de idei, opinii, informaţii şi soft între membrii săi.

Clubul Român de Calculatoare:

— asigură cadrul legal de organizare a posesorilor de calculatoare personale. (...)

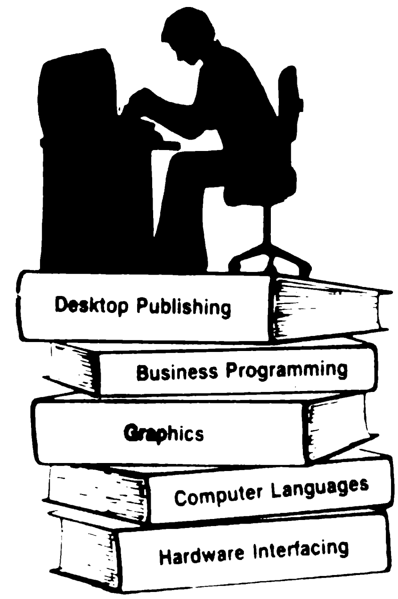
— creează bancă de programe.

— creează şi actualizează un sistem propriu de documentaţie. (...)

— apără interesele membrilor săi în relaţiile cu alte cluburi.

(...)

Art.17 — Prezentul statut intră în vigoare astăzi 28.01.1990.



UN CLUB DE CARE ERA NEVOIE !

CLUBUL ROMÂN DE CALCULATOARE

Nu am făcut decît să prezint cîteva articole din statutul clubului nostru. Cred că o mică prezentare a sa este binevenită. În 1987 cu ajutorul conducerii Casei de Cultură a Studenţilor a luat fiinţă un cerc care se dorea de microinformatică. Timid la început, compus doar din 7 membri, cercul îşi desfăşura activitatea duminica dimineaţa. Ne axam mai mult pe calculatoare tip Spectrum. Încet, încet, cercul şia mărit numărul de membri. Au apărut tot mai mulţi posesori de Commodore în detrimentul celor cu Spectrum. Au apărut şi „rarităţi": Atari 800, Atari 1024, Apple, Toshiba. Numărul de programe, atît distractive, cît şi utilitare, a crescut aproape logaritmic.

Cum era şi normal, după 22 decembrie, cercul s-a legalizat, devenind astăzi CLUBUL ROMÂN DE CALCULATOARE. Membri fondatori ai acestui club sînt: ing. Obretin Călin, ing. Gorodcov Mihaela, ing. Pîrlog Mircea, stud. Manasoiu Dan şi stud. Florean Cătălin.

În prezent sînt înscrise în club 124 persoane şi numărul lor creşte cu fiecare şedinţă. Sînt persoane inoculate cu acest „microb" al calculatorului folosit acasă, ingineri, studenţi, muncitori, copii, vîrstnici, adică majoritatea posesorilor de calculatoare personale din Bucureşti. Avem şi corespondenţi din ţară, membri ai clubului cu activitate permanentă.

Am reuşit în acest interval de 3 ani să strîngem o bogată documentaţie atît hard, cît şi soft. Specialiştii clubului, fără falsă modestie, pot face faţă oricărei probleme; de fapt acest lucru este cunoscut de un număr tot mai mare de posesori de hard ce au avut probleme de rezolvat.

Ce se poate face la o şedinţă a clubului? În primul rînd schimb de programe; avem peste 2 500 de programe atît de Spectrum cît şi Commodore, utilitare (turbo tape, turbo disc, procesoare de texte, soft pentru grafică şi muzică) dar şi programe distractive pentru toate genurile şi toate vîrstele. Din păcate la noi în ţară, majoritatea programelor ajung (sau ajungeau, pînă acum) cu o întîrziere de aproape un an faţă de data apariţiei lor în vest. Sperăm ca prin contactele stabilite de club cu cîteva firme şi cluburi din exterior termenul acesta să se micşoreze.

O altă preocupare a noastră este actualizarea sistemului de documentaţie, sistem accesibil oricărui membru al clubului. Pentru cei interesaţi, avem ultimele cataloage de hard ale firmelor Apple, Macintosh, IBM precum şi soft-ul acestora.

Dar, pe lîngă toate aceste preocupări de rutină, acum încercăm să iniţiem prima reţea de calculatoare personale (la domiciliu), reţea ce va utiliza liniile telefonice existente. Avem deja acordul forului competent, MODEM-ul este realizat, chiar legătura între două Commodore a fost experimentată, mai rămîn mici amănunte de protocol de pus la punct. Dorim

să creeăm o bancă de programe la care să poată avea acces orice membru CRC, de acasă, stînd comod în fotoliul din faţa televizorului propriu.

Pentru toţi prietenii, posesori sau neposesori ai unui calculator personal, care vor să ne contacteze pentru schimbul de programe, de idei, care vor să comunice preocupările lor din domeniu, care doresc să devină membrii CRC, îi rog să ne scrie pe adresa:

CLUBUL ROMÂN DE CALCULATOARE
P.O. Box 37-131
BUCUREŞTI,

precizînd în scrisoare tipul calculatorului cît şi lista de programe ce le oferă şi cea dorită în cazul schimbului de soft.

Pe cei din Bucureşti îi aşteptăm în fiecare sîmbătă la ora 11 la Casa de Cultură a Studenţilor, Calea Plevnei nr.61.

SOCIETATEA ROMÂNĂ DE INFORMATICĂ

În două adunări care au avut loc, în martie 1990, într-unul din marile amfiteatre ale Institutului Politehnic din Bucureşti, s-a constituit **Societatea Română de Informatică.** Au participat specialişti de la Politehnică, de la ASE şi de la Universitate, dar intenţia este de a fi atraşi în cadrul societăţii toţi cei care, indiferent de domeniul în care lucrează, pot beneficia de aportul gîndirii algoritmice şi utilizării calculatoarelor. S-a discutat un proiect de statut, s-a ales prin vot secret un consiliu de conducere şi s-a decis ca în noiembrie 1990 să aibă loc o nouă adunare generală. **Sediul Societăţii Române de Informatică este în sala EC106 de la Facultatea de Automatică a Institutului Politehnic Bucureşti (Splaiul Independenţei 313, sectorul 6), iar informaţiile privind modalitatea de a deveni membru al SRI se pot obţine la int. 320, la oricare dintre numerele de telefon 31.04.69, 31.40.10, 31.41.20.**

Scopul este promovarea ştiinţei calculatoarelor, a metodelor de producere, prelucrare, stocare şi transmitere a informaţiei şi stimularea educaţiei în domeniul informaticii, la toate nivelurile, de la copii pînă la adulţi. Societatea urmăreşte să contribuie la clarificarea rolului informaticii în societatea românească actuală, la studiul dezvoltării şi proiectării, realizării şi utilizării calculatoarelor şi a tehnicilor de prelucrare a informaţiei, la elaborarea metodelor şi programelor corespunzătoare. De asemenea, societatea îşi propune să faciliteze schimbul liber de idei, cunoştinţe şi informaţii din domeniul calculatoarelor, între diferite grupuri de interese din ţară şi din străinătate.

Dintre activităţile societăţii de informatică, menţionăm: ● elaborarea şi însuşirea unor norme de conduită profesională ● reprezentarea opiniilor, a punctelor de vedere şi a experienţei membrilor săi în probleme legate de evoluţia informaticii, în cadrul comisiilor de lucru pe probleme ● promovarea şi stimularea cercetărilor originale din domeniu prin organizarea de conferinţe, simpozioane, seminarii, expoziţii, concursuri profesionale şi premii acordate de societate ● implicarea acesteia în elaborarea strategiei de informatizare a societăţii româneşti ● instituirea unor relaţii de acces specifice membrilor săi la bibliotecile de specialitate şi alte centre de informare din ţară şi din străinătate ● stabilirea de contacte şi colaborări cu alte societăţi şi instituţii de specialitate ● organizarea de cursuri de specialitate şi tabere de instruire în domeniul informaticii ● stimularea introducerii metodelor, ideilor şi rezultatelor informaticii în şiinţele naturii şi în cele umaniste, în artă şi literatură.

Societatea de informatică va organiza, în cadrul ei, comisii de lucru pe probleme şi grupuri de interes. Un punct important îl constituie faptul că poate deveni membru al său orice persoană, indiferent de cetăţenie. Sperăm astfel să aducem în cadrul societăţii pe numeroşii informaticieni români care lucrează în străinătate.

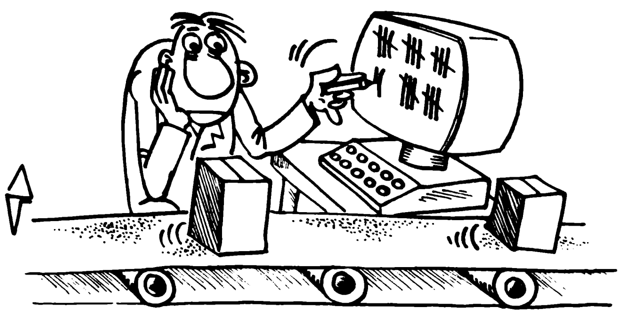
INFO
•sfaturi !
•previziuni !

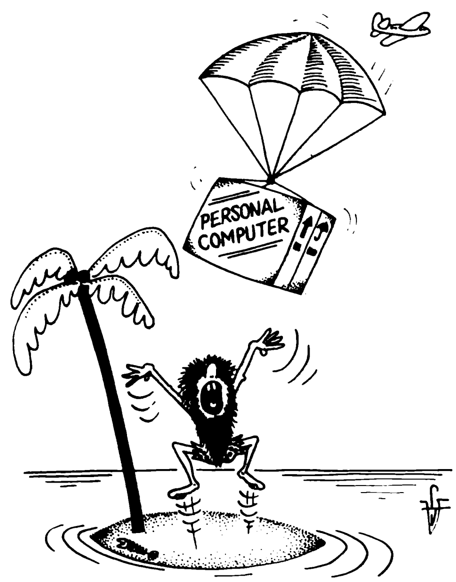


1. Calculatoarele tind să devină din ce în ce mai mici, aşa încît, dacă procesul continuă mereu, vor dispare cu... desăvîrşire.

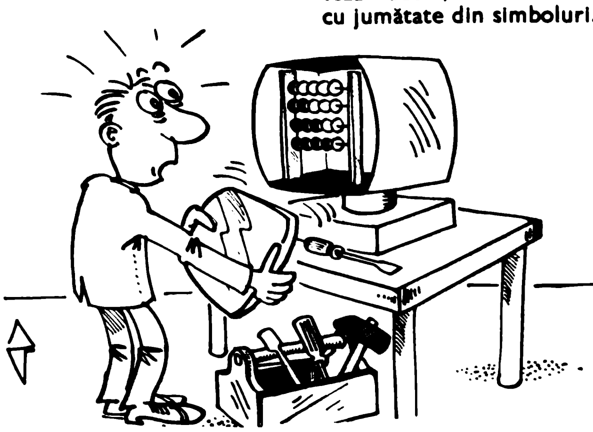
2. Perfecţionarea continuă a programării va duce la scrierea de programe care să-şi corecteze singure eroriLELE...

6. În Africa vor exista numai terminale display-color, cele alb/negru (a se citi alb pe negru) devenind interzise cu... desăvîrşire.





3. Evoluţia calculatoarelor, pe linie hardware, poate să ducă la apariţia ordinatoarelor unare, care oferă avantajul imediat: viteză dublă, lucrînd numai cu jumătate din simboluri.

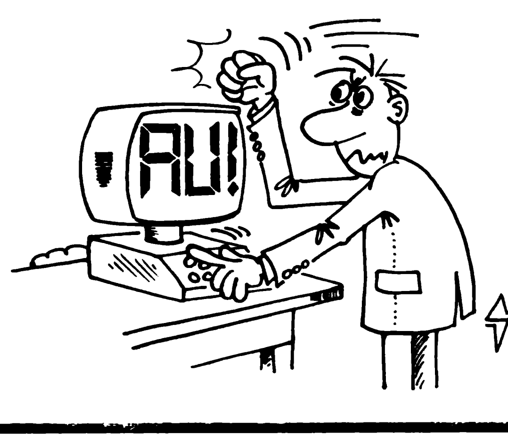


7. Atunci cînd păstrezi ceva în memoria calculatorului, fii prevăzător şi ţine minte unde... ai memorat.

8. Fii precaut în activitatea de programare deoarece un program face ce-i spui şi nu... ce-ai vrea să facă.

4. Se întrezăreşte, am zice de mai multă vreme, dispariţia FORTRAN-ului, dar credem că nu se realizează acest lucru pentru motivul că s-a propus a fi înlocuit de... MORTRAN.

5. Gurile rele (sau invidioşii) spun că o soartă similară o are şi fratele COBOL, căruia să-i ia locul MORBOL-ul, şi asta pentru a avea încă o aplicaţie a zicalei „să moară şi capra confratelui".



Umor

9. Multă, multă precauţie în ce priveşte fenomenul GIGO (gunoi pui, gunoi obţii) aplicat în programare.

10. Atenţie la detectarea şi corectarea erorilor unui program deoarece, în multe cazuri, după ce se efectuează o corecţie se deduce că, de fapt... era bine înainte.



11. Atenţie, fără lălăială în activitatea de programare, deoarece un program poate fi considerat depăşit înainte de a fi implementat.

12. Nu ştergeţi un fişier decît după ce v-aţi convins că... nu este al dumneavoastră.

13. Decît să vă întristaţi că nu ştiţi să apelaţi un program dintr-o bibliotecă, mai bine... miraţi-vă cum de a ajuns acolo.

14. Limbajul ASSIRIS nu a fost creat de asirieni şi nici SOCRATE de... Socrate. De aceea vă recomandăm să le trataţi ca atare.

Stelian NICULESCU

O posibilă discuţie...

Axiomă: toate persoanele prezente sînt excluse/incluse, după cum, respectiv, este vorba despre aspecte negative/pozitive sau invers, cum vă convine.

Vă rog să-mi permiteţi să nu mă prezint, cel puţin din trei motive:
1) n-aţi putea face acelaşi lucru;
2) nu se întîmplă nimic rău dacă acceptăm ideea că ne cunoaştem mai mult ori mai puţin ... mult;
3) un al treilea motiv poate fi găsit de oricare dintre dumneavoastră.

Şi ca să începem discuţia, vă propun ca aceasta să fie de tip n-log. Dacă n=1 cădem peste monolog, ceea ce nu preferăm, căci are o singură parte, ceea ce implică monotonie. În cazul n=2 apare dialogul, care are două părţi. Aşadar, nu sîntem toţi într-o parte (ca în cazul monologului). Şi ca să fim mai expliciţi, să considerăm că eu sînt într-o parte, iar dumneavoastră în cealaltă parte (sau, în general, în celelalte părţi, dacă n este mai mare ca 2).

Odată stabilit că discuţia este de tip n-log, se pune problema cine începe. Evident, cel mai simplu este să înceapă partea „într-o", adică eu. De ce? Simplu. Dacă aţi începe dumneavoastră, atunci ar apărea n-l lema. Cînd n=2 este clar, dar dacă n este mai mare sau egal cu 3 apare dilema, trilema ş.a.m.d. De aceea, vă rog să acceptaţi să încep eu, în felul acesta asigurînd şi continuarea şi sfîrşitul discuţiunii noastre, urmînd, totuşi, ca dumneavoastră să aveţi cuvîntul cînd ne întîlnim.

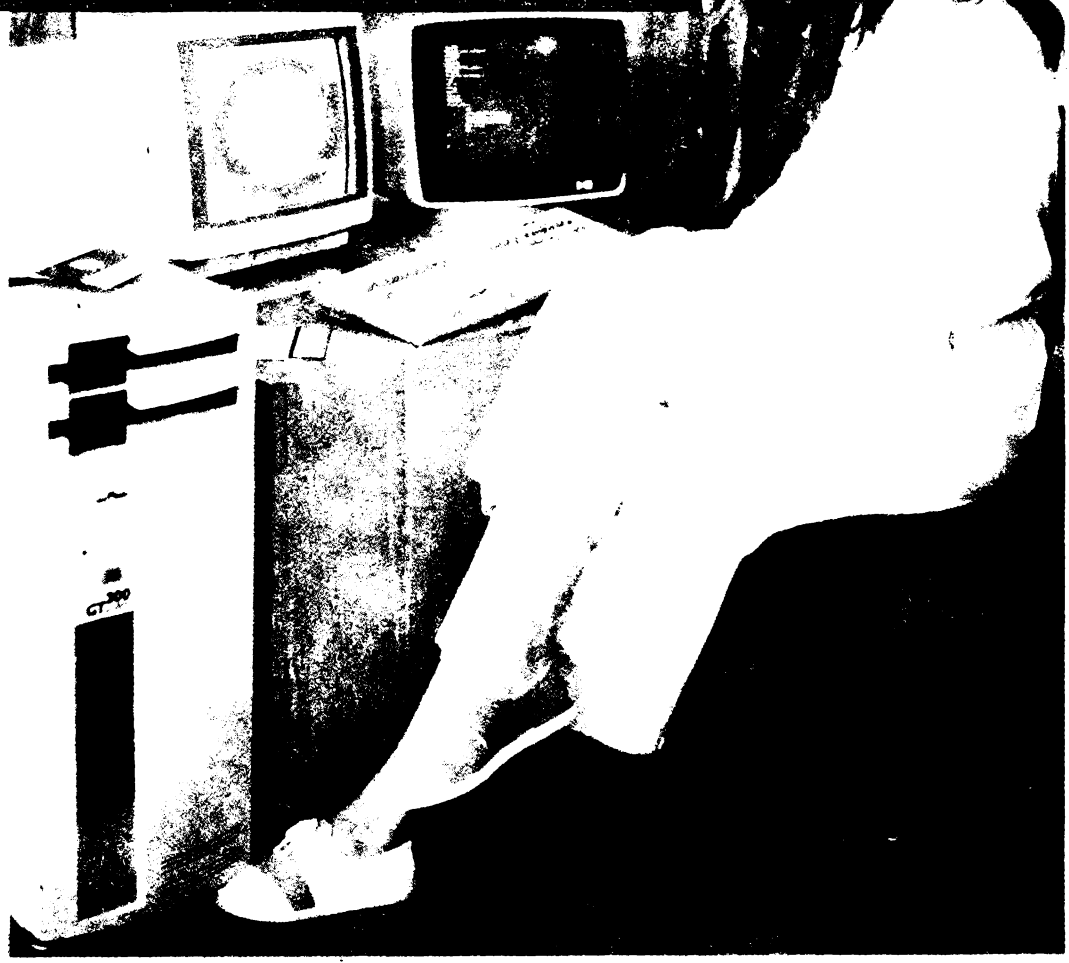
Şi ca să începem (de fapt, să continuăm discuţia), vă propun un model de „vreme" probabilă la o întîlnire prietenească, colegială ori cum doriţi dumneavoastră, cu referiri la ce fu şi la ce va fi în jurul momentului în discuţie, opţiunea noastră generală fiind aceea a bunei dispoziţii, ştiut fiind că viaţa fiecărula este mult prea finită.

În perioada ce o prognozăm, vremea calculatoarelor va fi în general plăcută, cu o atmosferă total favorabilă frumosului. Se vor semnala, cu totul şi cu totul izolat, unele înnourări şi/sau zîmbete „pe sub mustaţă". Unele averse intermitente vor duce la creşterea nivelului lichidelor, ceea ce va implica unele „încălziri" cu efecte plăcute, urmate (dar mai bine nu) de ceaţă locală, asociate cu fenomene izolate de vedere bidirecţională (în sensul de dublă).

Spre sfîrşitul intervalului ce-l prognozăm (din păcate mult prea scurt spre a fi pe măsura binelui şi bucuriilor ce vi le dorim cu toată sinceritatea momentului) va predomina buna dispoziţie locală particulară, eventual tulburată, cu totul izolat, de posibile averse locale provocate, de exemplu, de despărţiri, pe diverse fronturi... atmosferice locale.

În concluzie, timp predominant favorabil unui mediu ambiant plăcut, cu toate că în final pot să apară nuanţe nostalgice, dar care estimăm a fi grabnic trecătoare, fiind izolate.



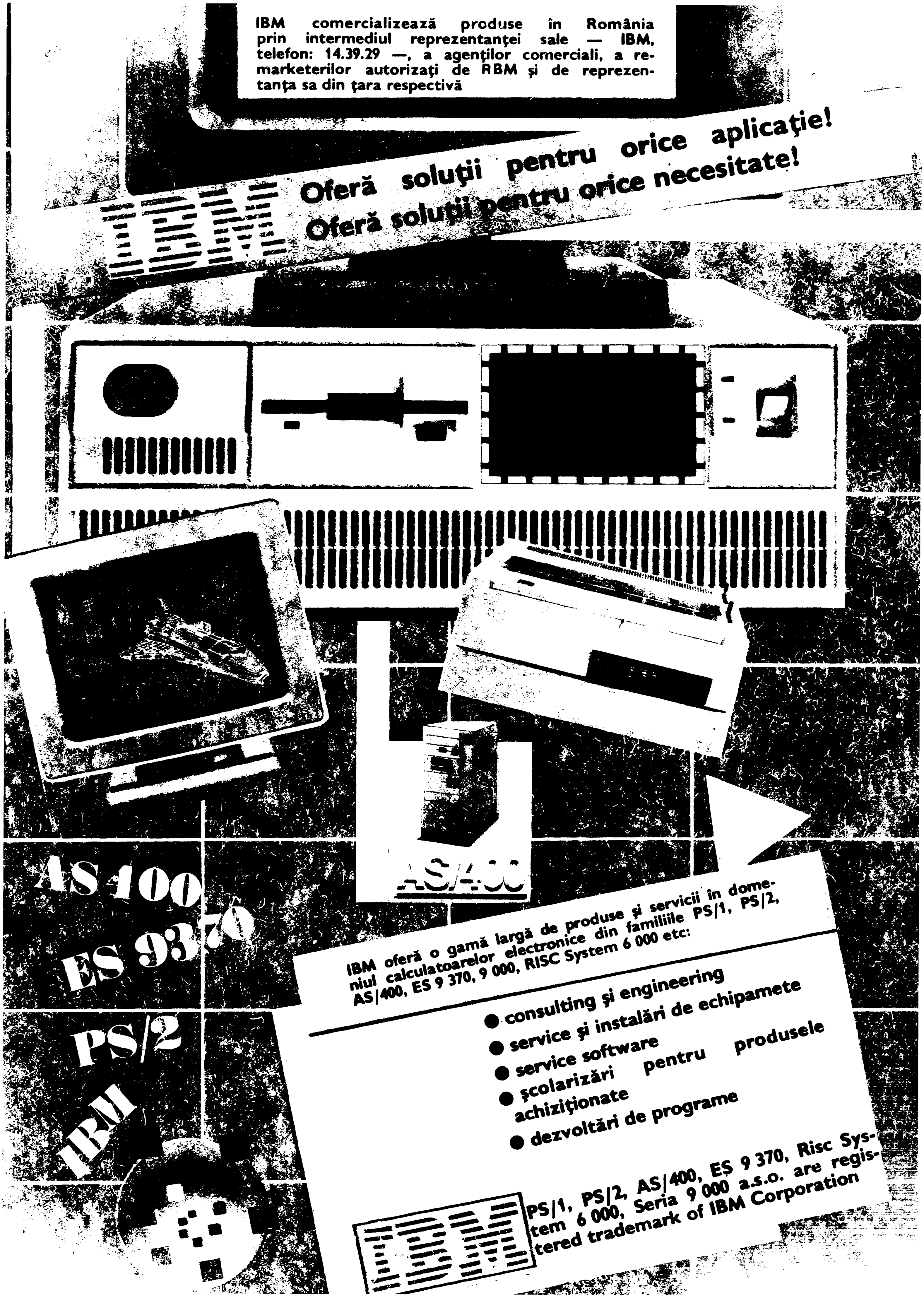


ÎNTREPRINDEREA DE CALCULATOARE ELECTRONICE pune la dispoziţia celor interesaţi o gamă completă de echipamente de tehnică de calcul pentru configurarea de aplicaţii personificate:

- **minicalculatoare 16/32 biţi, compatibile 100% cu modele PDP 11 şi VAX ale firmei DEC;**
- **microcalculatoare compatibile IBM PC/XT şi AT;**
- **echipamente periferice: discuri magnetice de pînă la 750 MBy, monitoare color grafice de înaltă rezoluţie, reţele de calculatoare, imprimante matriceale şi cu laser.**

Puterea unei REŢELE DE CALCULATOARE reprezintă siguranţa unei utilizări eficiente. În plus reţelele eterogene oferite de noi garantează dezvoltarea viitoare fără modificări în dotarea existentă.

ÎNTREPRINDEREA DE CALCULATOARE ELECTRONICE BUCUREŞTI
Str. Ing. G. Constantinescu nr. 2, sectorul 2, telefon: 88.22.95, telex 11 626 felix r, telefax: 88.78.20 felix r


IBM comercializează produse în România prin intermediul reprezentanţei sale — IBM, telefon: 14.39.29 —, a agenţilor comerciali, a re-marketerilor autorizaţi de RBM şi de reprezen-tanţa sa din ţara respectivă
IBM
Oferă soluţii pentru orice aplicaţie!
Oferă soluţii pentru orice necesitate!
AS/400
AS 400
ES 9370
PS/2
IBM
IBM oferă o gamă largă de produse şi servicii în dome-niul calculatoarelor electronice din familiile PS/1, PS/2, AS/400, ES 9 370, 9 000, RISC System 6 000 etc:
● consulting şi engineering
● service şi instalări de echipamete
● service software
● şcolarizări pentru produsele achiziţionate
● dezvoltări de programe
IBM
PS/1, PS/2, AS/400, ES 9 370, Risc Sys-tem 6 000, Seria 9 000 a.s.o. are regis-tered trademark of IBM Corporation